System User Guide

2012

This manual is designed for the technical user who maintains, administers, configures, and repairs the Phoenix system. This overview looks at the Phoenix software as a whole.

X.4.2.0

REVISED: 04/30/2012

ABM DATA SYSTEMS
896 SUMMIT STREET, SUITE 107
ROUND ROCK, TX 78664
(512) 388-3250

Table of Contents

How to Contact ABM Data Systems

Correspondence and Billing Address

ABM Data Systems
896 Summit Street, Suite 107
Round Rock, TX 78664
(512) 388-3250

Sales

ABM Sales hours are Monday through Friday, 8:00 a.m. to 5:00 p.m. Central Standard Time.
Sales Phone: **(800) 767-7067**
Sales Fax: **(512) 215-4110**
Sales Email: sales@abmsystemsllc.com
World Wide Web Address: www.abmsystemsllc.com

Technical Support

ABM Support hours are Monday through Friday, 8:00 a.m. to 5:00 p.m. Central Standard Time.
Support Phone: **(800) 729-4226**
(512) 388-3250
Support Fax: **(512) 215-4110**
Support Email: support@abmsystemsllc.com

This manual may help you resolve Signal Safe problems. Please contact our Technical Support Department if we can help in any way.

I. System

This manual is designed for the technical user who maintains, administers, configures, and repairs the Phoenix system. This overview looks at the Phoenix system as a whole.

A. Booting Up the Phoenix Computer

1. HASP Dongle

For Phoenix to run properly, a HASP dongle (hardware key) must be attached to a USB port. If the HASP dongle is not attached, after 14 days Phoenix will shut down or fail to start. See the ***Phoenix Installation Guide***.

2. Database

Depending on your Phoenix product, your database will be listed in Services as one of the following:

MS SQL Server
SQL Server (MSSQLSERVER)
A1 - MS SQL Server
A1 - SQL Server (MSSQLSERVER)

3. Services

When the Phoenix server is powered up, the SQL Server and the Phoenix Application Server (appsrv) services are started automatically. The Phoenix Application Server must be running for Phoenix Clients to login. The clients include Alarm Processing, Browser, Data Entry, Search, and Manager.

a) How to Verify Services Started

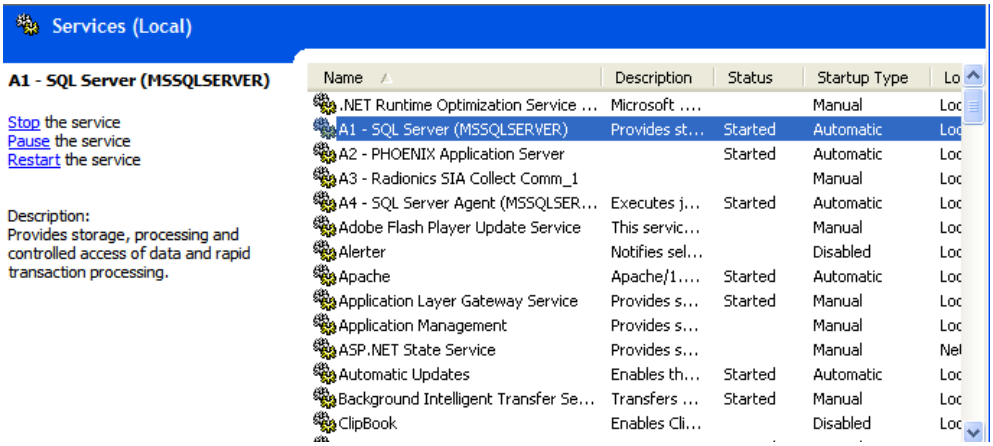
(1) For Windows XP OS or above

(a) *Click on the Start or Windows Icon in the Left hand corner of screen.*

(b) *Click on Control Panel, select Administrative Tools and then Services.*

(c) *After the services window opens, scroll through the list until you find either A1-MSSQLSERVER, A1-SQL SERVER (MSSQLSERVER), MSSQLSERVER, SQL SERVER (MSSQLSERVER)*

*Look in the Status Column and verify it says **Started**. If blank then the service is not started.*



(d) *Scroll through the list until you see either A1-PHOENIX Application Server or PHOENIX Application Server.*

Look in the Status Column and verify it says ***Started****. If it is blank then service is not started.*

b) How to Manually Start Services

(1) Correct Starting Order

(a) *Click on the Start or Windows Icon in the Left hand corner of screen.*

(b) *Click on Control Panel, select Administrative Tools and then Services.*

(c) *After the services window opens, scroll through the list until you find either A1-MSSQLSERVER, A1-SQL SERVER (MSSQLSERVER), MSSQLSERVER, SQL SERVER (MSSQLSERVER)*

Look in the Status Column and verify it is blank.

(d) *Right Click on the Database then select Start. Wait until the status turns to* ***Started.***

(e) *Scroll through list to find PHOENIX Application Server.*

(f) *Right Click then select Start. The status should change to* ***Started*** *once service is running.*

(g) *Scroll through list again and find the Receiver that is being used and do the same steps; right click select Start.*

(2) Correct Stopping Order

(a) *Correct stopping order is as follows:*

(i) Any receivers

(ii) PHOENIX Application Server

(iii) SQL Database

(b) *To stop each service above, select in order; Right click on that service, select Stop service and then wait until Status is blank before moving to the next service in line.*

(3) Trouble Shooting Service

If the Status column for a Service shows Started, but the Stop button is grayed out, then the Service is hung in an intermediate stage. This will require a reboot of the complete system to clear the hung service.

B. Setting up Receiver(s) to Run with Phoenix

Collects are the receiver driver programs in Phoenix that "collect" the information from the receiver and interpret it, breaking the signal string into the different fields of information that Phoenix needs to process the alarm.

NOTE: *It is important to setup your receivers to work with Phoenix early in the data entry process so that you can test the data entry before you enter too many records.*

1. Verifying Receiver Settings

Verify that the receiver is configured for automation software. Call the receiver manufacturer for assistance if you do not have the documentation showing how to setup the receiver you have. You can also ask ABM to see if they have a manual.

2. Correct Cables

There are different cable configurations that can be used for between the Phoenix Server and the receiver. Check with ABM Technical Support to see if there are any special pin-outs that is needed for your particular receiver and Phoenix configuration.

3. COM Settings

The following COM settings must be determined for the receiver:

Baud rate Parity Data bits
Stop Bits Flow Control

The receiver manufacturer can help you with determining the correct settings the receiver can use.

4. Testing COM Port

After the cables are hooked up and the COM settings are determined, test the COM port for connectivity. Depending on what version of Windows you are using, you either can use HyperTerminal or a program that replaces HyperTerminal.

> ***NOTE:*** *if using Windows 7 and above, HyperTerminal does not come with it. You can copy from Windows XP two files,* ***hypertrm.dll*** *and* ***hypertrm.exe****. Simply copy those files into any directory on your machine and it will work.*
> *You should be able to find* ***hypertrm.exe*** *in C:\ProgramFiles\WindowsNT and* ***hypertrm.dll*** *in C:\Windows\System32 on a Windows XP computer.*
> *You can also contact ABM to see if they have copies of those files to send to you.*

a) Using HyperTerminal

(if using something other than HyperTerminal, follow the directions for that program to test comport devices)

(1) Open HyperTerminal

Open the HyperTerminal program either by clicking on Start Menu, program, Accessories, HyperTerminal then click on HyperTerminal again. Or Click on the hpertrm.exe in the directory you copied the 2 files to from the NOTE above.

(2) Connection Description

In the Connection Description Window, fill in the Name field. You can use anything here. Example: Test

(3) Connect To Window

When the Connect To window opens, click on the Connect using: drop down and select the COM the receiver is connected to.

(4) COM Properties Window

In the COM# Properties dialog box, fill in the values for the serial settings as determined in ***Step 3, page 9***. Click on OK once done. (***Bit per second is the baud rate***)

(5) Test Signal

When the HyperTerminal window opens; send a signal through the receiver. The signal string should appear in the blank window. If the signal string looks correct, then go to ***step a) (6)***.

If the screen stays blank, no signal received, the problem may be in the cable. Verify the cable pin outs are correct or call the manufacturer for assistance.

If the packet string looks like gibberish (garbage), then problem could be the COM settings. Verify that the values programmed in the receiver are the same selected in HyperTerminal.

(6) Close HyperTerminal

HyperTerminal can be closed once correct signals start coming through. The session can also be saved for later use, if needed.

5. Verifying Collect Values

Verify the receiver settings are the same in the Phoenix Collect INI file as programmed into the receiver.

a) Finding Collect INI file

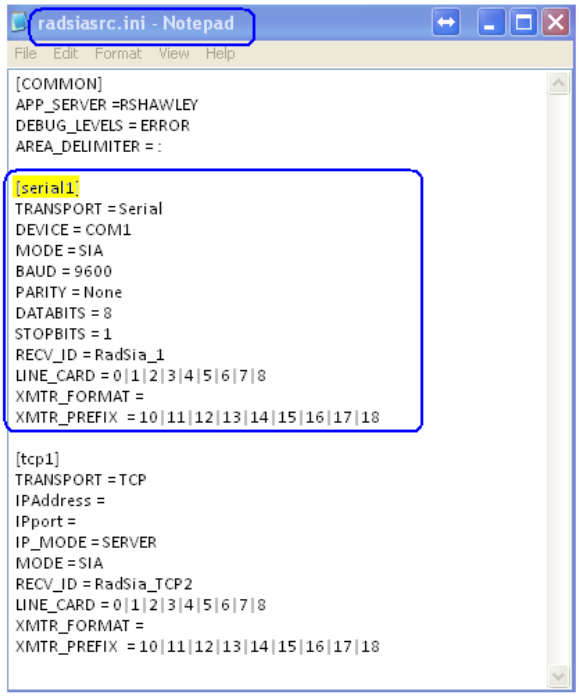
The Collect INI file is located: [*drive*]: \phoenix\profiles\collect
The collect file will be named as follows: [*collect_name*]src.ini
Example: ademcosrc.ini
Open the collect file with Notepad by double clicking.

b) Updating Collect INI file

(1) Port Section

Find the correct serial or TCP section to be updated.

Example:
Attaching a Radionics 6600 Receiver to Port 1 on the back of the computer. All changes would be done in the **[serial1]** section of the ini file.

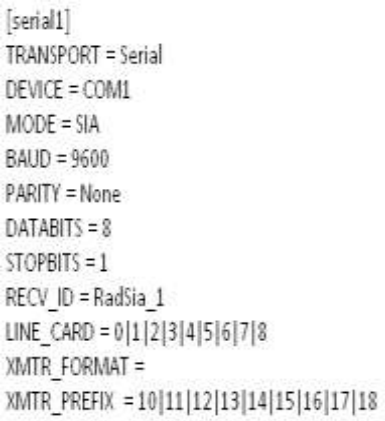


(2) Settings

There are several settings that can be changed so the receiver will work properly with Phoenix.

(a) *BAUD*

This is set to the same baud rate that the receiver is programmed.



(b) *PARITY*

This is set to the same parity rate that the receiver is programmed.

(c) *DATABITS*

This is set to the same parity rate that the receiver is programmed.

(d) *STOPBITS*

This is set to the same parity rate that the receiver is programmed.

(e) *RECV_ID*

Enter in how the receiver is to be named when coming in Phoenix. Can keep default naming or add own.

(f) *LINE_CARD*

Line cards are pre-configured for numeric's 0 – 8. With the new receivers, you can go up to 99. If you program them as alphas, change the LINE_CARD parameter accordingly. Line cards can also be programmed 00-99.

(g) *XMTR_PREFIX*

If you do not need to use transmitter prefixes, you can delete everything after the equal sign on the XMTR_PREFIX line.

If you are using rollover lines on multiple line cards, adjust XMTR_PREFIX accordingly.

Example: there are three rollover lines, so the first three (only ones being used) have the same prefix.

LINE_CARD = 00|01|02|03|04|05|06|07|08
XMTR_PREFIX = 10|10|10|13|14|15|16|17|18

c) Saving Updates

Save the collect ini file after all settings are completed.

6. Application Server

The Phoenix Application Server Service has to be running before the collect service will start. Prior to starting the Phoenix Application Server Service, verify the database is also running. Follow the steps below:

(a) *Click on the Start or Windows Icon in the Left hand corner of screen.*

(b) *Click on Control Panel, select Administrative Tools and then Services.*

(c) *After the services window opens, scroll through the list until you find either A1-MSSQLSERVER, A1-SQL SERVER (MSSQLSERVER), MSSQLSERVER, SQL SERVER (MSSQLSERVER)*

Look in the Status Column and verify it is blank. If the status shows Started, then skip to ***step (e)****.*

(d) *Right Click on the Database then select Start. Wait until the status turns to* ***Started.***

(e) *Scroll through list to find PHOENIX Application Server.*

(f) *Right Click then select Start. The status should change to* ***Started*** *once service is running.*

7. Collect Service

(a) *Click on the Start or Windows Icon in the Left hand corner of screen.*

(b) *Click on Control Panel, select Administrative Tools and then Services.*

(c) *After the services window opens, scroll through the list until you find either A1-MSSQLSERVER, A1-SQL SERVER (MSSQLSERVER), MSSQLSERVER, SQL SERVER (MSSQLSERVER)*

Look in the Status Column and verify it is started. If status shows ***Started****, skip to* ***step (e)****.*

(d) *Right Click on the Database then select Start. Wait until the status turns to* ***Started.***

(e) *Scroll through list to find* ***PHOENIX Application Server****.*

*Look in the Status Column and verify it is started. If status shows **Started**, skip to **step (g)**.*

(f) *Right Click then select **Start**. The status should change to **Started** once service is running.*

(g) *Scroll through list again and find the Receiver that is being used and do the same steps; right click select Start.*

(h) *When the Status column indicates **Started**, the Collect is running.*

8. Troubleshooting Problems

If the Collect fails to start, follow these steps to narrow down the problem:

a) Collect Hangs

If the Collect hangs (does not start within 10 – 15 seconds and both the start and stop buttons are dimmed), verify that the Application Server Service is started.

If the Application Server Service is started, stop it and reboot machine. Try restarting all services again in the proper order, see Correct Starting Order – Page 8, if the services did not start automatically.

b) Collect Won't Start

If all other Phoenix services are starting without a problem, except the Collect, try starting the Collect service from the command line.

(1) Command Prompt

Open the CMD Prompt by clicking on the Start (or Windows Icon), Run (click in Search), and type in CMD, then return.

(2) Command Line Statement

When the CDM window opens, type in the following statement and then Enter:

[collect_name.exe] -c -f [drive]:\phoenix\profiles\collect\[collect_name.ini] -s serial[#]

rad6500.exe -c -f c:\phoenix\profiles\collect\radsiasrc.ini -s serial3

(a) *If the blinking cursor goes below your statement and just blinks, this means the collect is started. There is something wrong with the Windows Service. Call ABM Tech Support for assistance.*

(b) *If an error message comes up, Call ABM Tech Support for assistance, there could be something wrong with the file or path for the Collect.*

c) Contact ABM Technical Support

If the collect still won't start or keeps hanging, call for assistance.

C. Setting the Language for Phoenix

Phoenix supports multiple languages. The language can be set to be used every time any user logs in or can prompt the user for the language each time they log in. The language displayed in each Phoenix application is controlled in the application's INI file.

1. Setting-Up INI's

a) Default Value

The default value displayed in the **[INIT]** section:
LANGUAGE=English-USA

As long as this parameter contains a value, Phoenix does not prompt users to select a language each time they log into the application.

b) Prompting for Value

To prompt users as they log in, delete the **English-USA** value so the parameter will only say:
LANGUAGE=

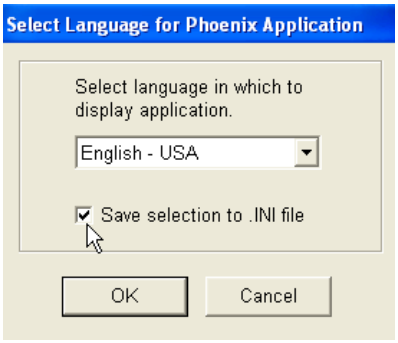
c) Changing Default Value

Follow the ***Steps in C1b***, and then save the INI file. Then log into the Phoenix application, when the Select Language Dialog Window pops up, select the language to be used.

To save the value, check the Save selection to .INI file, then click OK button. Now the new default language will come up every time the application opens.

d) Prompt for Language Every Time

Do not select the Save selection to .INI file before clicking on OK button in *Step c). Then the next time a user logs into the application they will be prompted to set the application to the language they want to see.*



2. Application INI's

The changes will need to be made in all INI's that are to have the language either changed or prompted. The INI files are located ***"\Phoenix\profiles"***.

The list is as follows:

Alarm.ini
Browser.ini
Maint.ini
Manual.ini
Search.ini

3. Field Name Translations

To modify the translation of field names that appear on the screen to reflect region-specific or dialect language differences, if other than English-USA is selected, you will need to modify the user-defined CFG file that is used by the Phoenix applications.

NOTE: DO NOT** change the **ABMmaster.cfg file**, only the **user.cfg

a) User CFG File

In the user configuration file, you may change three fields:

- **TableName**
- **Prompt**
- **StatusMsg**

b) Distributing CFG file

Once you have the user.cfg file set the way you want the prompts to show, save the file and copy to all clients that will have those settings. You can also have different user.cfg files for different operators.

Contact ABM Technical Support for assistance.

D. The User 'phoenix'

Phoenix ships with User ID ***998*** set up in the User table with a Login ID of "***phoenix***". It is configured as an administrative user.

***Caution** – this user is required for Phoenix administrative activities; **DO NOT delete it**. Contact ABM's Technical Support Department before you change any aspect of the **phoenix** user, such as password, etc.*

II. Location of Phoenix Components

The Phoenix system is installed in the Phoenix directory of the primary machine. Components of the system are stored in the following locations.

A. Phoenix\data

This directory holds the SQL Server database files, which includes the phoenix.mdf and phoenix.ldf. The security key file is also located in this directory.

B. Phoenix\profiles

This directory holds the Profile Initialization Files (*.ini) for Phoenix applications.

1. Phoenix\profiles\strings

This directory holds the language folders, one for each language supported by Phoenix. Within the language folders there will be files used by Phoenix applications and Phoenix Reporting. Those files are:

- Configuration files (*.cfg)
- Language files (*.str)
- Initialization files (*.INI)

2. Phoenix\profiles\collect

This directory holds the Profile Initialization Files (*.ini) for the Phoenix Collect applications, or receiver connection.

3. Phoenix\profiles\reports

This directory holds the pagesetup.ini used by Phoenix Reporting.

4. Phoenix\profiles\recurring_reports

This directory holds the reports defined as Recurring Reports in Phoenix Reporting.

C. Phoenix\bin

This directory holds the Phoenix Program executables and libraries.

D. Phoenix\tmp

This directory holds all the log files for the Phoenix Applications. ***Example***: alarm, browser, maint, manager, or any program logs for whoever logs into one of the Phoenix Applications. This directory also has the Log, Print and Mail directories.

1. Phoenix\tmp\log

This directory holds the log files for the Phoenix Application Server, Collects and User Logs. As each log expands a new log will form with a ***"name_#.log"*** this way the log files do not get over 1 Mg in size.

2. Phoenix\tmp\print

This directory holds all the print files that have been done in the Phoenix applications. ***Example:*** Printing an event ticket or doing a search in one of the tables. This is where those files are saved to either print later or to copy and email the print file. All files are saved in HTML format.

3. Phoenix\tmp\mail

This directory is where the mail files that are sent through Phoenix Mail are kept until ready to send to Outlook.

E. Inetpub\

This is the directory where the Phoenix Reporting files are located.

1. **Inetpub\scripts**
   This directory is where the Report Server (rptsrv.exe) program is located.

2. **Inetpub\wwwroot\phoenix**
   This is the directory where different Phoenix Reporting components are located.
   a) ***Inetpub\wwwroot\phoenix\images\ [language]***
   This is the directory where the image files (GIFs, etc.) used by the Phoenix Reporting System are located.

   b) ***Inetpub\wwwroot\phoenix\spool***
   This directory holds all the reports in the format assigned that have been run on the Phoenix Reporting System. These reports can be copied and emailed.

   c) ***Inetpub\wwwroot\phoenix\src\ [language]***
   This directory is where the HTML forms and JavaScript files used by the Phoenix Reporting System are located.

III. Phoenix Services in Windows

There are six services that Phoenix uses in order to work properly. Not everyone will have all of the listed services on their Phoenix System.

A. Phoenix Services

1. **Database Server Service**
   This service is configured to start automatically when the database software is installed. If this service is not started, the Phoenix Application Server Service will not be able to start.

2. **Appsrv – Phoenix Application Server Service**
   This service is configured to start automatically when the Phoenix Software is installed. This service has to be started prior to any of the Phoenix Applications can be logged into. Prior to starting this service the database service has to also be started.

3. **Collects – Receiver Server Service**
   This service is so the receiver(s) that will be connected to the Phoenix Server can send alarm information. The Collects that are part of the initial implementation of Phoenix are setup as services for you. For any collects purchased in the future, you may have to set up new comports and services that those comports.

4. **Watchdog – Phoenix UL Add-On Service**
   This service is only offered if purchased and if the central station is considered a UL Central Station.

5. Alarm Forwarding – Phoenix Add-On Service

This service will only be installed if the Add-On Module is purchased. It is configured when the module is installed.

6. Relay – Phoenix Tunneling Software

This service comes with the Phoenix Software and provides support for the Phoenix broadcasting messages across networks linked by a non-IGMP network device and for certain multi-NIC situations. Some setups do not require this service and will not be setup if not needed.

B. Phoenix Services File

The **PHXservices** file is used to define the Phoenix Services that are installed under the Service Control Manager and Registry. The information in this file is used by the **svcmgr** utility program when you add a Service.

1. Location

The PHXservices file is located: ***[drive]:\phoenix\profiles*** directory.

2. Sections/Settings

PHXservices file is broke down into **Sections** and **Settings**.

a) Sections

The Sections are the different services that can be created for Phoenix to use.

b) Settings

The Settings are the different variables in setting up the Service.

```
[APPSRV]
ServiceName      = appsrv
ServiceDisplayName = A2 - PHOENIX Application Server
ServiceType      = SERVICE_WIN32_OWN_PROCESS
StartMethod      =SERVICE_AUTO_START
ExecutablePath   = D:\Phoenix\bin\appsrv.exe
LoadOrderGroup   =
ID               =
Dependencies     =
Account          =
Password         =
Options          = -f D:\Phoenix\profiles\appsrv.ini


[ADEMCO_1]
ServiceName      = ademco_1
ServiceDisplayName = A3 - Ademco 685 Comm_1
ServiceType      = SERVICE_WIN32_OWN_PROCESS
StartMethod      = SERVICE_AUTO_START
ExecutablePath   = D:\Phoenix\bin\ademco.exe ademco_1
LoadOrderGroup   =
ID               =
Dependencies     =
Account          =
Password         =
Options          = -f D:\Phoenix\profiles\collect\ademcosrc.ini -s serial1
```

(1) ServiceName: {*name*}
This variable is the internal name used by the Service.

(2) ServiceDisplayName: {*name*}
This variable is the name displayed in the Services dialog box of the Service.

(3) ServiceType: {SERVICE_WIN32_OWN_PROCESS}
This variable is the type of Service.

(4) StartMethod: {SERVICE_AUTO_START, SERVICE_DEMAND_START}
This variable defines how the Service starts, either automatically or manually.

(5) ExecutablePath: {*path*}
This variable is the full path to the application program containing the Service.

(6) LoadOrderGroup: {*string*}
This variable is a string containing NULL or one of the load ordering groups from the Registry.

(7) ID: {*string*}
This variable is an ID value for the ordering group used.

(8) Dependencies: {*string*}
This variable is a pointer to an array of strings containing the names of Services that must start before this one.

(9) Account: {*string*}
This variable is the account the Service uses when it starts.

(10) Password: {*string*}
This variable is the password of the account.

(11) Options: {*string*}
This variable defines the options for the Service.

C. Phoenix Services Executable

The Phoenix install CD automates installing and deleting up to twelve Collects. When installing or deleting more than twelve Collects, use **svcmgr** as described in, ***"Manually Installing a Phoenix Service"*** page 21.

1. Location

The svcmgr is located: [***drive***]:\phoenix\profiles\

2. Manually Installing a Phoenix Service

The **svcmgr** utility program is used to install (register), delete (deregister), start, or stop a Phoenix Service using the Service Control Manager and the System Registry.

Example: when you purchase a new Collect from ABM, you use svcmgr to set it up as a Service.

When using **svcmgr**, you must input the values for several command line options.

Example: what action to perform, the exact path where the file is located, and the name of the file.

Option	***Definition***
-f	*file*
-i	*install*
-d	*delete*
-s	*start service*
-x	*stop service*

a) Install a Collect

To manually install a Collect, enter in the command shown below. Once this command is executed, the collect is included in the Services dialog box; it is available for starting and stopping.

Below example will install a Radionics (BOSCH) 6600 receiver on Com 2:

svcmgr -f [*drive*]:\phoenix\profiles\PHXservices -i rad6500_2

b) Delete a Collect

To manually delete a Collect, enter the command shown below. Once this command is executed, the Collect is removed from the Services dialog box; it is no longer available as a Service.

Below example will uninstall Radionics (BOSCH) 6600 receiver on Com2:

Svcmgr -f [*drive*]:\phoenix\profiles\PHXservices -d rad6500_2

IV. Profiles – Initialization Files

Many applications have the inherent ability to modify their run-time behavior at startup via the use of Profiles (also called Initialization files and Configuration files). Profiles contain variable data. By convention, many have a file extension of .ini; therefore they are often called INI files. The common extension for configuration files is .cfg.

Each INI file is composed of one to several sections containing initial startup values. Each section and its values apply to a specific component of the application.

The INI files are located in the **\phoenix\profiles** directory. You can view these files by opening them with a text editor. Some are the INI files are also in the \WINNT directory. The profiles version is permanent while the WINNT version can be deleted.

> ***Caution—****Open Initialization files in Notepad.* ***Do not*** *use word processing software such as* ***MSWord*** *or* ***WordPerfect*** *to edit* ***INI*** *files because they may corrupt the data.*
>
> ***Note—****any text string noted within braces,* ***{}****, is a variable. If more than one non-italicized variable is within the brackets, those variables are the exact verbatim options for the parameter. If there is a single italicized variable within the bracket, that string stands for a changeable value.*
>
> ***Helpful Hint —*** *If a parameter is explained in this manual, but does not exist in the applicable INI file on your system, you may add it to the INI file.*

A. Application Server

The Application Server's profile is composed of sections with each sections' name in upper case and enclosed by square brackets, [].

1. Appsrv INI File

a) [APPSRV] Section

This section describes the Application Server's startup settings.

The Appsrv section defines the values needed to:

- Configure the queue sizes of Shared Memory
- Define the database (ODBC), administrative user and password
- Define debug levels
- Define shared memory recovery option
- Define messenger broadcast addressing A typical

Appsrv section might look as follows:

```
[APPSRV]
ACTIVE_EVENT_QUEUE_SIZE = 10
PENDING_EVENT_QUEUE_SIZE = 50
WAITING_EVENT_QUEUE_SIZE = 50
EVENT_SIGNAL_QUEUE_SIZE = 100
WAIT_SIGNAL_QUEUE_SIZE = 20
HISTORY_SIGNAL_QUEUE_SIZE = 100
RAW_SIGNAL_QUEUE_SIZE = 100
CLIENT_QUEUE_SIZE = 25
USER_QUEUE_SIZE = 25
PHOENIX_INSTALLED_DEVICE = d:
DB_SERVER = server1
DB_USER = phoenix
DB_USER_PASSWORD = flight
```

```
DEBUG_LEVELS = ERROR
LOGIN_LEVELS = IN|OUT
USER_LEVELS = INSERT|UPDATE|DELETE
LOG_FILE_MAX_SIZE = 1000000
LOG_FILE_MODE = a+
USER_LOG_FILE_MAX_SIZE = 1000000
USER_LOG_FILE_MODE = a+
LOGIN_LOG_FILE_MAX_SIZE = 1000000
LOGIN_LOG_FILE_MODE = a+
RELOAD_SHARED_MEMORY = YES
NUMBER_SIGNAL_PROCESSORS = 2

LISTEN_MSG_ADDR = 234.0.0.9
RESPONSE_MSG_ADDR = 239
```

The APPSRV section is used to construct the Shared Memory Database. The queue sizes depicted generally represent the minimum number of signals or events expected to be in concurrent use. A process known as forecasting is used to adjust queue sizes as needed, but queue sizes will never be cast below the minimums noted here.

(1) ACTIVE_EVENT_QUEUE_SIZE = {##}

The ACTIVE_EVENT_QUEUE_SIZE defines an initial estimate of the minimum number of events that are likely to be active concurrently. An active event is an event that an operator is currently processing.

(2) PENDING_EVENT_QUEUE_SIZE = {##}

The PENDING_EVENT_QUEUE_SIZE defines an initial estimate of the minimum number of events that are likely to exist in the Pending Queue at any one time. The Pending Queue is the staging area for newly created events awaiting user action (processing) and for events that have been partially processed and placed back into Pending for further processing.

(3) WAITING_EVENT_QUEUE_SIZE = {##}

The WAITING_EVENT_QUEUE_SIZE defines an initial estimate of the minimum number of events awaiting further action. The user, as part of the event recall mechanism, can place events in this queue. When the user-defined time has expired, the event is moved back into the Pending Queue.

(4) EVENT_SIGNAL_QUEUE_SIZE = {##}

The EVENT_SIGNAL_QUEUE_SIZE defines an initial estimate of the minimum number of signals that can be placed in the Event Signal Queue. Each signal in this queue is associated with an event; there can be many signals associated with a single event.

(5) WAIT_SIGNAL_QUEUE_SIZE = {*##*}

The WAIT_SIGNAL_QUEUE_SIZE defines an initial estimate of the minimum number of signals related to events stored in the Signal Wait Queue. Certain signals require a counterpart signal to arrive pending further processing. In the interim, the initiating signal is placed in the Wait Queue for a defined time limit.

(6) HISTORY_SIGNAL_QUEUE_SIZE = {*##*}

The HISTORY_SIGNAL_QUEUE_SIZE defines the number of signals stored in the revolving signal queue. This queue size remains constant and is used to store signals as they are received by the Application Server. Once the queue is full, the next new signal is placed into the queue and the oldest signal is removed. This queue services signal/event requests from the Browser client.

(7) RAW_SIGNAL_QUEUE_SIZE = {*##*}

The RAW_SIGNAL_QUEUE_SIZE defines the initial estimate of the number of raw (preprocessed) signals received from the Collect. All such signals are stored in this queue as they are received to be retrieved and processed by the Signal Processors of the Application Server.

(8) CLIENT_QUEUE_SIZE = {*##*}

The CLIENT_QUEUE_SIZE defines an initial estimate of the minimum number of Phoenix clients. Clients, or Service Requestor Components (SRC), are processes that rely on having their requests serviced by Service Provider Components (SPC). The client queue contains information about each component of the Application Server. Components may be SPCs servicing SRC endpoints or system monitoring components.

(9) PHOENIX_INSTALLED_DEVICE = {drive}

The PHOENIX_INSTALLED_DEVICE defines the physical storage device (hard disk drive) on which Phoenix is installed. Other components of the Application Server reference this setting.

(10) DB_SERVER = {ODBC name}
DB_USER = {*user_name*}
DB_USER_PASSWORD = {*user_password*}

DB_SERVER, DB_USER, and DB_PASSWORD define the information required to gain access to the PHOENIX database. The PHOENIX database is identified by the name assigned by the DB_SERVER variable. The user ID defined by the DB_USER variable and password value associated with the

DB_USER_PASSWORD variable must be supplied before permission is granted to access the database.

(11) DEBUG_LEVELS = {*level*}

DEBUG_LEVELS determines the detail level of diagnostic message reporting. The messages are reported in a log file and are a useful tool for identifying error conditions. See "Debug Levels" on page 99 for an in-depth description of these options. If this parameter has no defined options, no messages of this type are written to the log file. If no parameter is defined, the log file remains empty. Any single option or combination of options may be used; use the pipe, |, separator between each option.

ERROR—reports any condition that is out of the ordinary.

TRACE—reports the entrance or invocation of a function.

INFO—reports an in-depth trace of the code being executed in a function.

Caution—*Leaving the INFO or TRACE parameters on for any period of time will cause the log file to become extremely large. Care should be taken when enabling these options.*

(12) LOGIN_LEVELS = {IN|OUT}

LOGIN_LEVELS determines the detail level in reporting the users' login and logout attempts. Each message is recorded in the **login.log** file, see "login.log" on page 105 for more information. If this parameter has no defined options, no messages of this type will be written to the log file. If no parameter is defined, the log file will remain empty. Login attempts are recorded for Alarm Processing, Data Entry, Manager and Search clients. Logout attempts are recorded for Alarm Processing clients.

(13) USER_LEVELS = {INSERT|UPDATE|DELETE}

USER_LEVELS determines the level of detail in reporting individual users' attempts to insert, update, or delete data from the database. Each message is recorded in the user.log file, see "user.log" on page 105 for more information. If this parameter has no defined options, no messages of this type will be written to the log file. If no parameter is defined, the log file will remain empty.

(14) LOG_FILE_MAX_SIZE = ######

LOG_FILE_MAX_SIZE allows you to specify the maximum size in bytes of the Debug log files. Default value is 1000000. As the initial log file (appsrv.log) reaches the size limit, Phoenix closes

it and creates a new file (appsrv__1.log) and writes to it. The digit suffix is incremented with each log file created.

(15) LOG_FILE_MODE = {a+, w+}

LOG_FILE_MODE allows you to specify how the Application Server initially opens and writes to the log. **a+** is the default. It opens a file for appending and creates the file if it does not exist. **w+** opens a file for writing. If the given file exists, its contents are destroyed.

(16) USER_LOG_FILE_MAX_SIZE = #######

LOG_FILE_MAX_SIZE allows you to specify the maximum size in bytes of the User log file. Default value is 1000000. As the initial log file (user.log) reaches the size limit, Phoenix closes it and creates a new file (user__1.log) and writes to it. The digit suffix is incremented with each log file created.

(17) USER_LOG_FILE_MODE = {a+, w+}

LOG_FILE_MODE allows you to specify how the Application Server initially opens and writes to the log. **a+** is the default. It opens a file for appending and creates the file if it does not exist. **w+** opens a file for writing. If the given file exists, its contents are destroyed.

(18) LOGIN_LOG_FILE_MAX_SIZE = #######

LOGIN_LOG_FILE_MAX_SIZE allows you to specify the maximum size in bytes of the Login log files. Default value is 1000000. As the initial log file (login.log) reaches the size limit, Phoenix closes it and creates a new file (login__1.log) and writes to it. The digit suffix is incremented with each log file created.

(19) RELOAD_SHARED_MEMORY = {YES, NO}

RELOAD_SHARED_MEMORY can be set to **YES** or **NO**. Invocation of the Application Server and an initial setting of **YES** will cause the contents of the Shared Memory Database to be reconstructed with any information that was part of shared memory at the time of the last Application Server termination. The contents of the Phoenix database are scanned for the existence of data that once resided in the Shared Memory Database. The Shared Memory Database being reconstituted may result in the delayed operation of components such as Browser. When the setting is changed to a **NO** setting it will not reconstruct shared memory.

***Caution** – If this parameter is set to **NO**, shared memory is not reconstructed. If a crash occurs, any unprocessed signal in shared memory will not be available for processing.*

(20) NUMBER_SIGNAL_PROCESSORS = {##}
NUMBER_SIGNAL_PROCESSORS defines the number of components of the Application Server which retrieve raw signals from the raw signal queue and process them to generate events. Setting this to an excessive number degrades performance. The system default is generally optimal.

(21) LISTEN_MSG_ADDR = {*###.#.#.#*}
The LISTEN_MSGR_ADDR defines the Class D internet address used by the Phoenix Application Server for its broadcast messaging service.

Caution—do not *change this value without instructions from ABM's Technical Support Department. Doing so could* ***disable*** *the Phoenix Messenger Service, thereby causing Phoenix to malfunction. The acceptable Class D address range is 224.0.0.0 to 239.255.255.255.*

(22) RESPONSE_MSG_ADDR = {*##*}
The RESPONSE_MSG_ADDR defines a Class D address component which Phoenix uses to construct multicast broadcast response messages.

Caution—do not *change this value without instructions from ABM's Technical Support Department. Doing so could* ***partially disable*** *the Phoenix Messenger Service, thereby causing Phoenix to malfunction. The acceptable Class D address range is 224 to 239.*

b) [RELAY]Section
This section covers settings used by the Messaging Relay component, if it is installed. A typical Relay section might look as follows:

```
[RELAY]
MESSAGE_RELAY = YES
RELAY_MESSAGEQ_SIZE = 6
```

(1) MESSAGE_RELAY = {YES, NO}
MESSAGE_RELAY indicates if the Message Relay component of the Application Server is active.

(2) RELAY_MESSAGEQ_SIZE = {#}
The RELAY_MESSAGEQ_SIZE defines the initial estimate of the number of messages addressed to other components of Phoenix that the Relay Component is responsible for broadcasting. The Relay component periodically checks the

message queue for any broadcast messages and performs any necessary actions.

c) *[FORCEASTING] Section*

This section covers settings used by the shared memory forecasting and resizing algorithm.

The settings in the Forecasting section are used by the Phoenix Queue Monitor component to forecast and resize shared memory queues as required. Each queue defined under the [APPSRV] section uses different parameters for the purposes of forecasting shared memory demand and therefore resizing. PREDICT_INTERVAL, PREDICT_SUPPLY, PREDICT_BUCKETS, and PREDICT_SENSITIVITY are the variables defined for each of the shared memory queues.

***Caution—do not** change these values without instructions from ABM's Technical Support Department.*

A typical Forecasting section might look as follows:

```
[FORECASTING]
CLIENT_PREDICT_INTERVAL = 6
CLIENT_PREDICT_SUPPLY = 2
CLIENT_PREDICT_BUCKETS = 60
CLIENT_PREDICT_SENSITIVITY = .2
 ACTIVE_EVENT_PREDICT_INTERVAL = 6
ACTIVE_EVENT_PREDICT_SUPPLY = 10
ACTIVE_EVENT_PREDICT_BUCKETS = 60
ACTIVE_EVENT_PREDICT_SENSITIVITY = .2
PENDING_EVENT_PREDICT_INTERVAL = 6
PENDING_EVENT_PREDICT_SUPPLY = 10
PENDING_EVENT_PREDICT_BUCKETS = 60
PENDING_EVENT_PREDICT_SENSITIVITY = .2
WAITING_EVENT_PREDICT_INTERVAL = 6
WAITING_EVENT_PREDICT_SUPPLY = 10
WAITING_EVENT_PREDICT_BUCKETS = 60
WAITING_EVENT_PREDICT_SENSITIVITY = .2
EVENT_SIGNAL_PREDICT_INTERVAL = 6
EVENT_SIGNAL_PREDICT_SUPPLY = 10
EVENT_SIGNAL_PREDICT_BUCKETS = 60
EVENT_SIGNAL_PREDICT_SENSITIVITY = .2
WAIT_SIGNAL_PREDICT_INTERVAL = 6
WAIT_SIGNAL_PREDICT_SUPPLY = 10
WAIT_SIGNAL_PREDICT_BUCKETS = 60
WAIT_SIGNAL_PREDICT_SENSITIVITY = .2
```

(1) {Queue}_PREDICT_INTERVAL = {#}
This parameter defines the frequency of observations in minutes, that is, the period of time between acquiring sample queue sizes.

(2) {Queue}_PREDICT_SUPPLY = {#}
This parameter defines a multiplier determining the minimum amount reserved based upon the predicted demand. This is the number of entries in intervals, that is, the queue contents determined in intervals per minute.

Note—*the client queue is not volatile as the other queues, therefore the CLIENT_PREDICT_SUPPLY is usually a smaller number than the other queues.*

(3) {Queue}_PREDICT_BUCKETS = {#}
This parameter defines the number of samples to retain for forecasting; each sample is the quantity of entries in the queue at that time it was polled.

(4) {Queue}_PREDICT_SENSITIVITY = {. #}
This parameter defines the percentage in a decimal value within which the forecasted value will be used to resize the queue.

Example: this parameter is set to .2 (20%). When the queue reaches 80% of capacity (within 20% of full capacity) the queue will be resized.

d) [LISTEN]Section

This section covers settings relating to the Application Server's connection service.

The Listen section defines settings used for the Listen component, which is responsible for the servicing of connection requests. Clients wishing to address the Application Server may do so by issuing a request to the well-known LISTEN TCP/IP port defined in the LISTEN_MSG_ADDR field. The Listen component responds by forwarding the TCP/IP address of the service being requested.

The typical Listen section might look like this:

```
[LISTEN]
KEEP_ALIVES_POLLING_INTERVAL        = 0
LISTEN_MSGQ_SIZE                    = 6
```

(1) KEEP_ALIVES_POLLING_INTERVAL = {#}

The KEEP_ALIVES_POLLING_INTERVAL parameter is not yet implemented.

(2) LISTEN_MSGQ_SIZE = {##}

The LISTEN_MSGQ_SIZE parameter defines an initial estimate of the number of messages addressed to the Listen component of the Application Server. The Listen component periodically checks the message queue for any broadcasted messages and performs any necessary actions.

e) [SYSTEM_MONITOR] Section

This section covers settings relating to Application Server system monitoring.

The System Monitor section defines settings used in monitoring system behavior.

> ***Note—****Minutes are entered in whole and fractions of a minute. For example, 1.25 equals one minute and 15 seconds.*

The typical System Monitor section might look like this:

```
[SYSTEM_MONITOR]
SYSMON_MSGQ_SIZE          =  6
SHMEM_POLLING_MINUTES     =  1.0
TRIGGER_TIME_LIMIT        =  480.0
CLEAN_EXPIRED_SCHEDULES   =  NONE
PENDEVT_AGE_LIMIT_MINUTES =  5.0
PENDEVT_DEADMAN_MINUTES   =  10.0

WATCHDOG_MONITORING       =  NO

COLLECT_POLLING_INTERVAL  =  120
BROWSER_POLLING_INTERVAL  =  120
MAINT_POLLING_INTERVAL    =  120
ALARM_POLLING_INTERVAL    =  120
MANAGER_POLLING_INTERVAL  =  120
WATCHDOG_POLLING_INTERVAL =  120
```

The SYSMON_MSGQ_SIZE parameter defines an initial estimate of the number of messages addressed to the System Monitor component of the Application Server. The System Monitor component periodically checks the message queue for any broadcast messages and performs any necessary actions.

(1) SHMEM_POLLING_MINUTES = {*#.#*}

The SHMEM_POLLING_MINUTES parameter defines the interval in which the System Monitor attempts to perform its tasks. If time to run equals 2 minutes and the interval equals 10 minutes, System Monitor will wait 8 minutes before running again. If time to run equals 14 minutes and interval = 10 minutes, System Monitor will wait the full 10 minutes before running again. System Monitoring immediately performs its tasks upon starting the Application Server.

System Monitor tasks include:

- Dongle Check: Validates the Phoenix security file and checks to see if a HASP dongle is connected to a USP port
- File System Integrity Check: performs a read operation on the database and a read/write operation on the hard drive.
- Return expired Wait Events to Pending
- Checks Pending Events against aging parameters
- Checks Active Events against aging parameters
- Checks that the Relay process is connected (if using Relay)
- Polling monitor: checks status of connected clients
- Reminder check: generates reminders
- Check disk space: warns if low on disk space (default setting performs this once every 24 hrs)
- Check Signal Table: checks to see if the ABMsignal table is purging regularly (default setting performs this once every 24 hours)
- Purge ABMsignal: deletes old items from ABMsignal table

(2) PENDEVT_AGE_LIMIT_MINUTES = {*#. #*}

This parameter defines the acceptable length of time in which an event can remain in the Event Pending Queue without having being assigned to an operator. Once an event has exceeded its aging limit, Alarm Processing operators are notified via the Phoenix messaging service.

(3) PENDEVT_DEADMAN_MINUTES = {#. #}

This parameter defines acceptable length of time which an event can remain in the Pending Event Queue without having being assigned to an operator. The events which meet this criteria will have the sigtype_class column assigned to “DeadMan”. The Phoenix Add-On Alarm Forwarding will then alert defined contact of the situation.

(4) ACTIVEENT_AGED_MINUTES = {#.#}

This parameter defines acceptable length of time which an event can remain in the Active Event Queue without having been updated by the operator assigned to the event. Once and event has exceeded its aging limit, Alarm Processing operators are notified via the Phoenix messaging service.

(5) WATCHDOG_MONITORING = {YES, NO}

The WATCHDOG_MONITORING parameter enables or disables Watchdog support through Appsrv. It is only effective if the Watchdog Add-on product is installed.

COLLECT_POLLING_INTERVAL = {#}
BROWSER_POLLING_INTERVAL = {#}
MAINT_POLLING_INTERVAL = {#}
ALARM_POLLING_INTERVAL = {#}
MANAGER_POLLING_INTERVAL = {#}
WATCHDOG_POLLING_INTERVAL = {#}

The POLLING_INTERVAL for COLLECT, BROWSER, MAINT, ALARM, MANAGER, and WATCHDOG define the interval in which the System Monitor component verifies that each of the mentioned components is operating effectively. If the component has not interacted with the Application Server within the Polling Interval, the System Monitor forces an acknowledgment from the relevant components. If the component has interacted with the Application Server within the Polling Interval, the System Monitor does not force an acknowledgment when it cycles.

A setting of **0** (zero) specifies that System Monitor forces an acknowledgment each time it wakes up to perform its tasks. A setting of **5** specifies that System Monitor request an acknowledgment from a component if it has not voluntarily checked in within 5 minutes. So if SHMEM_POLLING_MINUTES is set to 1 minute, the component(s) are checked on the fifth time that the System Monitor wakes up. Intervals may also be set in fractions of minutes (seconds). An entry of .25 means fifteen seconds and an entry of .5 (1/2 minute) translates to 30 seconds.

f) [SITE_MONITOR] Section

This section covers settings relating to open/close monitoring.

The Site Monitor section defines settings used in open and close monitoring. Open signals are sent from a panel when a Contact disarms the alarm system to gain access into the site. A close signal is sent from a panel when the Contact arms the system.

The typical Site Monitor section might look like this:

```
[SITE_MONITOR]
PERFORM_SITE_MONITORING = YES
OPENCLOSE_POLLING_MINUTES= 5.0
KEEP_ALIVES_POLLING_INTERVAL = 0
SITEMON_MSGQ_SIZE = 6
```

(1) PERFORM_SITE_MONITORING = {YES, NO}

The PERFORM_SITE_MONITORING parameter tells the Application Server if open/ close monitoring should take place. A setting of **NO** will keep the Site Monitor component from running. A setting of **YES** allows Site Monitor to begin monitoring for opening and closing signals.

Caution—*this parameter must be set to* ***YES*** *for the system to perform this type of* ***Monitoring****.*

(2) OPENCLOSE_POLLING_MINUTES = {*#. #*}

The OPENCLOSE_POLLING MINUTES parameter defines the interval in which the Site Monitor component runs to perform its tasks. If time to run equals 2 minutes and the interval equals 10 minutes, Site Monitor will wait 8 minutes before running again. If time to run equals 14 minutes and interval = 10 minutes, Site Monitor will wait the full 10 minutes before running again. One note of importance, upon initial Application Server startup the Site Monitor component is forced to sleep for the defined OPENCLOSE_POLLING_MINUTES. This is done to allow all the necessary application server components to get started without competition for resources.

(3) KEEP_ALIVES_POLLING_INTERVAL = {#}

The KEEP_ALIVES_POLLING_INTERVAL parameter is not yet implemented.

(4) SITEMON_MSGQ_SIZE = {*#*}

The SITEMON_MSGQ_SIZE parameter defines an initial estimate of the number of messages addressed to the Site Monitor component of the Application Server. The Site Monitor component periodically checks the message queue for any broadcasted messages and performs any necessary actions required of it.

g) [SITE_TESTMONITOR] Section

This section covers settings relating to panel test (heartbeat) monitoring.

The Site Test Monitor section defines settings used in the monitoring of panel test signals. These signals may be automated or manual. An automated signal would be sent in from the panel at a defined interval of time. These signals may be thought of as a heartbeat or a panel pulse and is meant to notify the monitoring station that, although no activity has occurred for the day, the panel is functioning properly.
A manual generation of a test signal is initiated by a guard service that goes from site to site making sure visually that everything is secured. The guard typically trips and resets the panel at a specified time.

A typical Site Test Monitor section might look like this:

```
[SITE_TESTMONITOR]
PERFORM_SITE_TESTMONITORING=YES
TESTMONITOR_POLLING_MINUTES = 10.0
SITETESTMON_MSGQ_SIZE = 6
```

(1) PERFORM_SITE_TESTMONITORING = {YES, NO}

The PERFORM_SITE_TESTMONITORING parameter tells the Application Server if test monitoring should take place. A setting of **NO** will keep the Site Test Monitor component from running. A setting of **YES** will allow the Site Test Monitor component to begin monitoring for test signals.

***Caution** – this parameter must be set to **YES** for the system to perform this type of monitoring.*

(2) TESTMONITOR_POLLING_MINIUTES = {#. #}

The TESTMONITRO_POLLING_MINUTES parameter defines the interval in which the Site Test Monitor component runs to perform its tasks. If time to run equals 2 minutes and the interval equals 10 minutes, Test Monitor will wait 8 minutes before running again. If time to run equals 14 minutes and interval = 10 minutes, Test Monitor will wait the full 10 minutes before running again. One note of importance, upon initial Application Server Startup the site test monitor component is forced to sleep for the defined TESTMONITOR_POLLING_ MINUTES. This is done to allow all the necessary application server components to start without competition for resources.

(3) SITETESTMON_MSGQ_SIZE = {#}

The SITETESTMON_MSGQ_SIZE parameter defines an initial estimate of the number of messages addressed to the Site Test Monitor Queue. The Site Test Monitor component periodically checks the message queue for any broadcasted messages and performs any necessary actions required of it.

h) [QUEUE_MONITOR] Section

This section covers settings relating to the monitoring of the sizes of the shared memory queues.

The Queue Monitor section defines settings used in the monitoring of the Shared Memory database. The Queue Monitor component uses information provided in the APPSRV and FORECASTING sections of the appsrv.ini file. Refer back to those sections for information regarding queue size and forecasting issues.

A typical Queue Monitor section might look like this:

```
[QUEUE_MONITOR]
PERFORM_QUEUE_MONITORING = YES
QUEUEMON_MSGQ_SIZE = 6
QUEUEMON_POLLING_MINUTES = 6.0
```

(1) PERFORM_QUEUE_MONITORING = {YES, NO}

The PERFORM_QUEUE_MONITORING parameter tells the Application Server if the component which monitors Queue sizes should run. A setting of **NO** will keep the Queue Monitoring component from running. A setting of **YES** will allow the component to begin monitoring the utilization of memory related to the queues.

***Caution** – this parameter must be set to **YES** for the system to perform this type of monitoring. A **NO** value may result in the loss of event data.*

(2) QUEUEMON_MSGQ_SIZE = {#}

The QUEUEMON_MSGQ_SIZE parameter defines an initial estimate of the number of messages addressed to the Queue Monitor component of the Application Server. The Queue Monitor component periodically checks the message queue for any broadcasted messages and performs any necessary actions required of it.

(3) QUEUEMON_POLLING_MINUTES = {*#. #*}

The QUEUEMON_POLLING MINUTES parameter defines the interval in which the Queue Monitor component runs to perform its tasks. This component does not sleep initially as does Site Monitor or Test Monitor. It is important that Queue Monitor begin taking snapshots of queue utilization as soon as possible to ensure that the queue sizes are adequately forecast.

i) [COLLECT] Section

The Collect section defines settings for the Collect SPC, which provides service and signal interpretation for the Collect Service Requestor Component.

A typical Collect section might look like this:

```
[COLLECT]
COLLECT_MSGQ_SIZE = 6
```

(1) COLLECT_MSGQ_SIZE = {*#*}

The COLLECT_MSGQ_SIZE parameter defines an initial estimate of the number of messages addressed to the Collect component of the Application Server. The Collect component periodically checks the message queue for any broadcasted messages and performs any necessary actions required of it.

j) [BROWSER] Section

The Browser section defines settings for the Browser SPCs with the sole task of servicing the requests of Browser SRCs.

A typical Browser section might look like this:

```
[BROWSER] BROWSE_MSGQ_SIZE = 6
```

(1) BROWSE_MSGQ_SIZE = {*#*}

The BROWSE_MSGQ_SIZE parameter defines an initial estimate of the number of messages addressed to the Browser component of the Application Server. The Browser component periodically checks the message queue for any broadcasted messages and performs any necessary actions required of it.

k) [ALARM_PROCESSING] Section

The Alarm Processing section defines settings for the Alarm Processing SPCs, which have the sole task of servicing the requests of the Alarm Processing SRCs.

A typical Alarm Processing section might look like this:

```
[ALARM PROCESSING]
AP_MSGQ_SIZE = 6
```

(1) AP_MSGQ_SIZE = {#}

The AP_MSGQ_SIZE parameter defines an initial estimate of the number of messages addressed to the Alarm Processing component of the Application Server. The Alarm Processing component periodically checks the message queue for any broadcasted messages and performs any necessary actions required of it.

l) [MAINTENANCE] Section

The Maintenance section defines settings for the Maintenance SPCs (the Data Entry user interface). The sole task of this component is to service the requests from the Maintenance SRCs.

The typical Maintenance section might look like this:

```
[MAINTENANCE]
MAINT_MSGQ_SIZE = 6
```

(1) MAINT_MSGQ_SIZE = {#}

The MAINT_MSGQ_SIZE parameter defines an initial estimate of the number of messages addressed to the Maintenance component of the Application Server. The Maintenance component periodically checks the message queue for any broadcasted messages and performs any necessary actions required of it.

m) [MANAGER] Section

The Manager section defines settings for the Manager SPCs. This component performs certain system functions such as:

- The facilitation of access to the Shared Memory Database
- The service of requests from Manager SRCs
- The service of requests to view, insert, modify and delete entries in the Shared Memory Database
- Phoenix application shutdown
- Component messaging

In addition, the Manager component provides a simple SQL interface to the Shared Memory database.

Note—*restrict this component to Administrative users.*

A typical Manager section might look like this:

[MANAGER]
MANAGER_MSGQ_SIZE = 6

MANAGER_MSGQ_SIZE = {#}
The MANAGER_MSGQ_SIZE parameter defines an initial estimate of the number of messages addressed to the Maintenance component of the Application Server. The Maintenance component periodically checks the message queue for any broadcasted messages and performs any necessary actions required of it.

2. alarm.ini—INI File for Alarm Processing

The Alarm INI file is used to define session configuration information for the Alarm Processing application (such as default sizes). When a user makes a change in Alarm Processing, the relevant screen parameters are recorded in this file. The next time an operator logs into Alarm Processing, the application reads this file and adjusts the session or screen settings appropriately. These settings are in effect each time the operator logs in until the settings are changed.

The Alarm INI is organized into sections with each section having a relevant heading. The following examines each section and its associated parameters:

a) Common Section

This section covers settings necessary to the startup of the Alarm Processing component.

The typical Common section might look like this:

[COMMON]
APP_SERVER = server1
BROADCAST_PORT = 8123
ApplicationProfile = user.cfg
DEBUG_LEVEL = ERROR
LOG_FILE_MAX_SIZE = 1000000
;INTERFACE_ADDRESS = 172.0.0.0

(1) APP_SERVER = {*hostname*}

The APP_SERVER parameter defines the internet hostname of the machine where the Application Server (appsrv) resides.

(2) BROADCAST_PORT = {####}

This parameter defines the port segment of the application server's LISTEN components IP address. This parameter should not be changed.

(3) ApplicationProfile = {*path*}

The ApplicationProfile parameter sets the location of the appropriate configuration file. This variable value indicates the user CFG file that supplies database parameters for Alarm Processing.

(4) DEBUG_LEVELS = {*level*}

The DEBUG_LEVELS parameter defines what type of diagnostic information is written to the log file. The variable value may be one or more of the following; however, it is most likely to be **ERROR**. Any single option or combination of options may be used; use the pipe, |, separator between each option.

ERROR—reports any condition that is out of the ordinary.
TRACE—reports the entrance or invocation of a function.
INFO—reports an in-depth trace of the code being executed in a function.

Caution—*Leaving the INFO or TRACE parameters on for any period will cause the log file to become extremely large. Care needs taken when enabling these options.*

(5) LOG_FILE_MAX_SIZE = ######

LOG_FILE_MAX_SIZE allows you to specify the maximum size in bytes of the Debug log files. Default value is 1000000. As the initial log file (appsrv.log) reaches the size limit, Phoenix closes it, creates a new file (appsrv.log_1), and writes to it. The digit suffix is incremented with each log file created.

(6) LOG_FILE_MODE = {a+, w+}

LOG_FILE_MODE allows you to specify how the Application Server initially opens and writes to the log, **a+** is the default. It opens a file for appending and creates the file if it does not exist. **w+** opens a file for writing. If the given file exists, its contents are destroyed.

(7) INTERFACE_ADDRESS

The INTERFACE_ADDRESS parameter is necessary to facilitate Phoenix messaging when the client is run on a remote network connected to the local (server) network though non-IGMP enabled devices, such as RAS or a modem connection. This situation requires that the Relay product be installed and runs as a service on one node of each remote network segment. The parameter value must be the exact IP address of the machine on which the client is running. Do not set this parameter for a client run on the local (server) network.

This parameter normally is preceded by a ";" character, which indicates it is a comment and therefore not if effect. When the leading semi-colon (;) is removed and the actual TCP/IP address of the machine the application is running on is entered, the machine can receive Phoenix broadcast messages through a RAS or modem connection as they are forwarded by the Relay product.

b) Init Section

This section contains information regarding the initialization of the Alarm Processing component.

The typical Init section might look like this:

```
[INIT]
loginid = administrator
MAX_WAIT_TIME = 43200
MAX_ROWS = 1000
HIGH_PRIORITY = GETALL
EVENTUPDATE_PRIORITY = EQUAL
EXPERT_MODE = FALSE
LANGUAGE = ENGLISH-USA
```

(1) loginid = {*user_id*}

The loginid parameter contains the Login ID of the last operator who logged in to Alarm Processing. The default is blank until an operator logs in, at which time the user Login ID is written here. The next time a user logs in; that Login ID appears in the Login ID field in the Login dialog box.

(2) MAX_WAIT_TIME

The MAX_WAIT_TIME parameter is the longest time in minutes that an operator can put an Event in Wait. (It is specific to the workstation or the operator using roaming profiles.)

(3) MAX_ROWS

The MAX_ROWS parameter defines the maximum number of rows that display in Queries.

(4) HIGH_PRIORITY = {GETALL, EQUAL, HIGHER}

The HIGH_PRIORITY parameter specifies the criteria for notifying an operator who is working an Event when new Events enter the system.

(a) ***GETALL*** *notifies for every new Event.*

(b) ***EQUAL*** *notifies if the signal has an equal or higher priority than the current Event processed.*

(c) ***HIGHER*** *notifies if the signal has a higher priority than the current Event processed.*

(5) EVENTUPDATE_PRIORITY = {GETALL, EQUAL, HIGHER, NONE}

The EVENTUPDATE_PRIORITY parameter specifies the criteria for notifying an operator about additional signals attaching to an Event that is in Pending or placed in Wait. Phoenix notifies regardless of whether the operator is currently working an Event.

(a) ***GETALL*** *notifies for every new signal.*

(b) ***EQUAL*** *notifies if the signal has an equal or higher priority than the Event currently being worked.*

(c) ***HIGHER*** *notifies if the signal has a higher priority than the Event currently being worked.*

(d) ***ONE*** *does not notify for any signal.*

(6) EXPERT_MODE = {TRUE, FALSE}

The EXPERT_MODE parameter allows you to define the workstation as being in "expert mode", which eliminates some of the confirmation dialog boxes in Alarm Processing, effectively minimizing the number of key strokes required to process Events. The default value is **FALSE**, which means all confirmation dialog boxes appear in Alarm Processing. When users are thoroughly familiar with Alarm Processing, you can eliminate some confirmation dialog boxes by setting the value to **TRUE**.

(7) AUTO_HISTORY = {0, 1}

When value is set to "1", History window will open automatically each time an event is dropped for processing.

(8) AUTO_ATTACHMENTS = {0, 1}

When set to a "1," the attachment dialog will open automatically when there are attachments on the transmitter or up the hierarchy whenever an event drops for processing.

(9) LANGUAGE = {ENGLISH-USA, SPANISH, PORTUGUESE}
The **LANGUAGE** parameter allows you to define the language to use on the workstation.

c) TAPI Section

This section covers the set up of the telephony interface. The typical TAPI section might look like this:

[TAPI]
Usemodem=False
PHONE = 123-4567
Usemodem = {True, False}

(1) UseModem
The **Usemodem** parameter determines whether TAPI (Telephony Application Programming Interface) is utilized by the system.

(2) PHONE
The PHONE parameter indicates the phone number where the operator can be contacted**.**

d) View Sizes Section

This section covers the screen set up of the Alarm Processing component.

Caution—*these settings are updated by the system and should NOT be changed by a user for any reason.*

The typical View Sizes section might look like this:

[VIEW SIZES]
Instruction=23.24
Contact=17.16
log=57.46
Location=42.68

(1) Instructions = {##.##} - Contact = {##.##} - Log = {##.##} - Location = {##.##}
These parameters define the width, height, and placement of elements on the Alarm Processing screen.

3. browser.ini—INI File for Browser

The browser.ini file is used to record user configuration information as well as information needed for the Browser application (such as default column sizes, default column names, default window sizes). When an operator makes a change in Browser, the parameters are recorded in this file. The next time any operator logs into Browser on the same machine, Phoenix reads this file and adjusts the screen settings appropriately. The settings are in effect each time an operator logs

in until the settings are changed again. The Browser profile is organized into sections with each section having a relevant heading. The following examines each section and its associated parameters:

This section covers settings necessary to the startup of the Browser component.

The typical Common section might look like this:

```
[COMMON]
APP_SERVER = server1
ApplicationProfile = user.cfg
DEBUG_LEVELS = ERROR|TRACE|INFO
LOG_FILE_MAX_SIZE = 1000000
LOG_FILE_MODE = a+
;INTERFACE_ADDRESS = 172.0.0.0
```

(1) APP_SERVER = {hostname}

The APP_SERVER parameter defines the internet hostname of the machine where the Application Server (Appsrv) resides.

(2) ApplicationProfile = {*path*}

The ApplicationProfile parameter identifies the location of the INI file that defines the actual text for each field on the screen. Both Browser and Data Entry should reference the same user CFG file.

(3) DEBUG_LEVELS = {*level*}

The DEBUG_LEVELS parameter defines what type of information is written to the log file. The variable value may be one or more of the following; however, it is most likely to be **ERROR**. Any single option or combination of options may be used; use the pipe, (|), separator between each option.

(a) ***ERROR****—reports any condition that is out of the ordinary.*

(b) ***TRACE****—reports the entrance or invocation of a function.*

(c) ***INFO****—reports an in-depth trace of code being executed in a function.*

Caution—*Leaving the INFO or TRACE parameters on for any period of time will cause the log file to become extremely large. Care should be taken when enabling these options.*

(4) LOG_FILE_MAX_SIZE = ######

LOG_FILE_MAX_SIZE allows you to specify the maximum size in bytes of the Debug log files. Default value is 1000000. As the initial log file (browser.log) reaches the size limit, Phoenix closes it and creates a new file (browser_1.log) and writes to it. The digit suffix is incremented with each log file created.

(5) LOG_FILE_MODE = {a+, w+}

LOG_FILE_MODE allows you to specify how the Application Server initially opens and writes to the log. ***a+*** *is the default. It opens a file for appending and creates the file if it does not exist.* ***w+*** *opens a file for writing. If the given file exists, its contents are destroyed.*

(6) INTERFACE_ADDRESS = ###.###.###.###

The INTERFACE_ADDRESS parameter is necessary to facilitate Phoenix messaging when the client is run on a remote network connected to the local (server) network though non-IGMP enabled devices, such as RAS or a modem connection. This situation requires that the Relay product be installed and run as a service on one node of each remote network segment. The parameter value must be the exact IP address of the machine on which the client is running. Do not set this parameter for a client run on the local (server) network.

This parameter is normally preceded by a ";" character, which indicates it is a comment and therefore not if effect. When the leading semi-colon (;) is removed and the actual TCP/ IP address of the machine Browser is running on is entered, the machine can receive Phoenix broadcast messages through a RAS or modem connection.

b) Signal Queue Section

This section covers settings applicable to the signal queues in the Browser component.

The typical Signal Queue section might look like this:

[SIGNAL QUEUE]
COLWIDTHS=55,80,60,60,60,40,40,150,90,90,70,75

(1) COLWIDTHS = {##,##,##,##,##,##,##,##,##,##,##,##}

The COLWIDTHS parameter defines the actual horizontal screen space used to display each specific column value. The column widths are set in the graphical user interface and stored in this file to be used the next time the user logs on.

c) Event Queue Section

This section covers settings applicable to the event queues in the Browser component.

The typical Event Queue section might look like this:

[EVENT QUEUE]
COLWIDTHS=40,80,60,80,40,40,60,150,90,90,80

(1) COLWIDTHS = {##,##,##,##,##,##,##,##,##,##,##}

The COLWIDTHS parameter defines the actual horizontal screen space used to display each specific column value. The column widths are set in the graphical user interface and stored in this file to be used the next time the user logs on.

d) Queue Bar Section

This section covers settings applicable to the Event Traffic Indicators in the Browser component.

The typical Queue Bar section might look like this:

[QUEUE BAR]
NOTICE=3
ATTENTION=5
Bucket1=Fire Alarm(101,102,103,104)
Bucket2=Duress(119)
Bucket3=Medical(65,105)
Bucket4=Burglary(96)
Bucket5=Open/Close(1,2,50,51,53,62,66,68)
Bucket6=Supervisory(106)
Bucket7=Equipment(88,89,90,91,92,93,94,95)
Bucket8=Other(0)

(1) NOTICE = {#}

The NOTICE parameter defines the number of items in a queue which will change the bucket's severity level indicator color from green to yellow.

(2) ATTENTION = {#}

The ATTENTION parameter defines the number of items in a queue which will change the bucket's severity level indicator color from yellow to red.

(3) Bucket {#} = {label} (sigcat#, sigcat#, ...)

The Bucket parameters correspond to the Event Traffic Indicators in Browser. You can change the labels on the screen by altering the bucket name here. Each time an event-gen-

erating signal is received with a Sigcat ID matching a number in parentheses, Phoenix updates the corresponding Event Traffic Indicator field on the Browser screen. If a signal's Sigcat ID does not correspond with one of the buckets, it is applied to the bucket. A signal's Sigcat is defined in the Sigtype record.

e) Init Section

This section contains information used when Browser is opened.

The typical Init section might look like this:

```
[INIT]
loginid=administrator
EXPERT_MODE = FALSE
LANGUAGE = ENGLISH-USA
```

(1) loginid = {*user_id*}

The loginid parameter contains the Login ID of the last operator who logged in. The default is blank until an operator logs in, at which time that Login ID is written here. The next time a user logs in that Login ID will appear in the Login ID field in the Login dialog box.

(2) EXPERT_MODE

The EXPERT_MODE parameter is not yet implemented in Browser.

(3) LANGUAGE = {ENGLISH-USA, SPANISH, PORTUGUESE}

The LANGUAGE parameter allows you to define the language to be used on the workstation.

f) General Section

This section covers settings which apply to the Browser component universally.

The typical General section might look like this:

```
[GENERAL]
 REFRESHRATE=15
```

(1) REFRESHRATE = {#}

The **REFRESHRATE** parameter sets how often (in seconds) the Browser screen is refreshed. Any new signals or Events which have entered the system are added to the display each time the screen is refreshed.

Caution—*A short refresh rate may dramatically impact system performance especially when two or more Browser components are open.*

Do not change the following sections without the assistance of ABM's Technical Support Department; since these are dynamically assigned settings. When an operator changes the size of a window etc., the changes are written to the variables, therefore preserving the workstation's settings. The next time an operator logs in on that workstation, the screen retains the same appearance.

g) Active Views Section

This section covers which Queues display by default in the Browser component.

Caution—these settings are updated by Phoenix. Do not change them for any reason.

The typical Active Views section might look like this:

[ACTIVE VIEWS]
Active Event=Active
Pending Event=Active
Waiting Event=Inactive
Event Signal=Inactive
Waiting Signal=Active
History Signal=Inactive

(1) {*Queue*} = {Active, Inactive}

The Queue parameters define which Queues default as active the next time the Browser component is started.

h) {Queue} Section

This section covers the attributes of each displayed queue.

The [{*Queue*}] section defines the screen settings for each individual Queue (i.e., Active Event, Pending Event, Waiting Event, Event Signal, Waiting Signal, and History Signal). These sections contain parameters detailing: whether the Queue is displayed on the screen, where on the screen it displays, the specific placement of its columns, etc. None of these fields is user tailorable.

The typical [{*QUEUE*}] section might look like this:

[History Signal]
Flags=0
WindowState=1
MinXPos=0
MinYPos=656

MaxXPos=-4
MaxYPos=-23
WindowTop=0
WindowBottom=226
WindowLeft=0
WindowRight=1119
COLWIDTHS=55,120,120,60,60,40,150,90,90,70,65

4. maint.ini—INI File for Data Entry

The maint.ini file records user configuration information as well as information needed for the Data Entry application (such as default column sizes, default column names, default window sizes, default text colors, and default background colors). When a user changes these parameters in Data Entry, the parameters are recorded in this file. The next time an operator logs into Data Entry on the same machine, Phoenix reads this file and adjusts the screen settings accordingly. The settings are in effect each time an operator logs in until the settings are changed again.

The maint.ini is organized into sections with each section having a relevant heading. The following examines each section and its associated parameters:

a) Common Section

This section covers settings necessary to the start-up of the Data Entry component.

The typical Common section might look like this:

[COMMON]
APP_SERVER = server1
DB_USER = phoenix
DB_USER_PASSWORD = flight
ApplicationProfile = user.cfg
DEBUG_LEVELS = ERROR|TRACE|INFO
LOG_FILE_MAX_SIZE = 1000000
LOG_FILE_MODE = a+
;INTERFACE_ADDRESS = 172.0.0.0

(1) APP_SERVER = {*hostname*}

The APP_SERVER parameter defines the internet hostname of the machine where the Application Server (Appsrv) resides.

(2) DB_USER = {*user_name*} / DB_USER_PASSWORD = {*user_password*}

DB_USER, and DB_PASSWORD define the information required to gain access to the PHOENIX database. The user ID defined by the DB_USER variable and password value associated with the DB_USER_PASSWORD variable must be supplied before permission is granted to access the database.

(3) ApplicationProfile = {*path*}

The ApplicationProfile parameter identifies the location of the INI file shared with Browser which lists the actual text of each item on the Browser and Data Entry screens and the name of the database item they relate to. Both Browser and Data Entry should point to the same file, user.cfg.

(4) DEBUG_LEVELS = {*level*}

The DEBUG_LEVELS parameter defines what type of information is written to the log file. The variable value may be one or more of the following; however, it is most likely to be **ERROR**. Any single option or combination of options may be used; use the pipe, |, separator between each option.

(a) ***ERROR***—*reports any condition that is out of the ordinary.*

(b) ***TRACE***—*reports the entrance or invocation of a function.*

(c) ***INFO***—*reports an in-depth trace of code being executed in a function.*

Caution—*Leaving the INFO or TRACE parameters on for any period of time will cause the log file to become extremely large. Care should be taken when enabling these options.*

(5) LOG_FILE_MAX_SIZE = ######

LOG_FILE_MAX_SIZE allows you to specify the maximum size in bytes of the Debug log files. Default value is 1000000. As the initial log file (maint.log) reaches the size limit, Phoenix closes it and creates a new file (maint_1.log) and writes to it. The digit suffix is incremented with each log file created.

(6) LOG_FILE_MODE = {a+, w+}

LOG_FILE_MODE allows you to specify how the Application Server initially opens and writes to the log. **a+** is the default. It opens a file for appending and creates the file if it does not exist. **w+** opens a file for writing. If the given file exists, its contents are destroyed.

(7) INTERFACE_ADDRESS

The INTERFACE_ADDRESS parameter is necessary to facilitate Phoenix messaging when the client is run on a remote network connected to the local (server) network though non-IGMP enabled devices, such as RAS or a modem connection. This situation requires that the Relay product be installed and run as a service on one node of each remote network segment. The parameter value must be the exact IP address of the machine on which the client is running. Do not set this parameter for a client run on the local (server) network.

This parameter is normally preceded by a ";" character, which indicates it is a comment and therefore not if effect. When the leading semi-colon (;) is removed and the actual TCP/IP address of the machine the application is running on is entered, the machine can receive Phoenix broadcast messages through a RAS or modem connection.

b) Init Section

This section contains information regarding the initialization of the Data Entry component.

The typical Init section might look like this:

```
[INIT]
loginid=administrator
maxrows=1000
EXPERT_MODE = FALSE
LANGUAGE = ENGLISH-USA
```

(1) loginid = {*user_id*}

The loginid parameter contains the Login ID of the last operator who logged in. The default is blank until an operator logs in, at which time that Login ID is written here. The next time a user logs in, that Login ID appears in the Login ID field in the Login dialog box.

(2) maxrows = {#}

This parameter defines the maximum number of rows returned as a result of a Query of the database. Setting this parameter too small limits the number of records you are able to Query in Data Entry. Setting it too large may dramatically impact system performance by allowing the selection of too many records.

(3) EXPERT_MODE = {TRUE, FALSE}

The EXPERT_MODE parameter allows you to define the workstation as being in "expert mode", which eliminates some of the confirmation dialog boxes in Data Entry. The default value is FALSE, which means all confirmation dialog boxes appear. When users are thoroughly familiar with Data Entry, you can eliminate some confirmation dialog boxes by setting the value to TRUE.

(4) LANGUAGE = {ENGLISH-USA, SPANISH, PORTUGUESE}

The LANGUAGE parameter allows you to define the language to be used on the workstation.

Do not change the following sections without the assistance of ABM's Technical Support Department since these are dynamically assigned settings. When an operator changes the size of a window or column, etc., the changes are written to these variables, therefore preserving the workstation's settings. The next time any user logs in on that workstation, the screen retains the same appearance.

5. managersrc.ini—INI File for Manager

This file is used by the System Manager SRC. Among other things, the contents of this file define the internet host location of the Application Server and set the Manager SRC debug level.

a) Manager Section

This section contains information regarding the System Manager executable.

The typical Manager section might look like this:

```
[MANAGER]
APP_SERVER = server1
DEBUG_LEVELS = ERROR|TRACE|INFO
;INTERFACE_ADDRESS = 172.0.0.0
```

(1) APP_SERVER = {server_name}

The APP_SERVER parameter defines the internet hostname of the machine where the Application Server (Appsrv) resides.

(2) DEBUG_LEVELS = {level}

*The DEBUG_LEVELS parameter defines what type of information is written to the log file. The variable value may be one or more of the following; however, it is most likely to be **ERROR**. Any single option or combination of options may be used; use the pipe, |, separator between each option.*

(a) ***ERROR***—*reports any condition that is out of the ordinary.*

(b) ***TRACE***—*reports the entrance or invocation of a function.*

(c) ***INFO***—*reports an in-depth trace of code being executed in a function.*

(3) INTERFACE ADDRESS

The INTERFACE_ADDRESS parameter is necessary to facilitate Phoenix messaging when the client is run on a remote network connected to the local (server) network though non-IGMP enabled devices, such as RAS or a modem connection. This situation requires that the Relay product be installed and run as a service on one node of each remote network segment. The parameter value must be the exact IP address of the machine on which the client is running. Do not set this parameter for a client run on the local (server) network.

This parameter is normally preceded by a ";" character, which indicates it is a comment and therefore not if effect. When the leading semi-colon (;) is removed and the actual TCP/ IP address of the machine the application is running on is entered, the machine can receive Phoenix broadcast messages through a RAS or modem connection.

6. manual.ini—INI File for Manual Signal

This file is used by the Manual Signal SRC. Among other things, the contents of this file define the internet host location of the Application Server and set the Manual Signal SRC debug level.

a) Common Section

This section contains information regarding the Manual Signal executable.

The typical Common section might look like this:

```
[COMMON]
APP_SERVER = server1
ApplicationProfile = user.cfg
DEBUG_LEVELS = ERROR
EXPERT_MODE = FALSE
LANGUAGE = ENGLISH-USA
```

(1) APP_SERVER = {*server_name*}

The APP_SERVER parameter defines the internet hostname of the machine where the Application Server (Appsrv) resides.

(2) ApplicationProfile = {*path*}

The ApplicationProfile parameter identifies the location of the INI file shared with Browser which lists the actual text of each item on the Browser and Data Entry screens and the name of the database item they relate to. Both Browser and Data Entry should point to the same file, user.cfg.

(3) DEBUG_LEVELS = {*level*}

The DEBUG_LEVELS parameter defines what type of information is written to the log file. The variable value may be one or more of the following; however, it is most likely to be **ERROR**. Any single option or combination of options may be used; use the pipe, |, separator between each option.

(a) ***ERROR****—reports any condition that is out of the ordinary.*

(b) ***TRACE****—reports the entrance or invocation of a function.*

(c) ***INFO****—reports an in-depth trace of code being executed in a function.*

Caution—Leaving the INFO or TRACE parameters on for any period of time will cause the log file to become extremely large. Care should be taken when enabling these options.

(4) EXPERT_MODE = {TRUE, FALSE}

The EXPERT_MODE parameter allows you to define the workstation as being in “expert mode”, which eliminates some of the confirmation dialog boxes in Data Entry. The default value is FALSE, which means all confirmation dialog boxes appear. When users are thoroughly familiar with Data Entry, you can eliminate some confirmation dialog boxes by setting the value to TRUE.

(5) LANGUAGE = {ENGLISH-USA, SPANISH, PORTUGUESE}

The LANGUAGE parameter allows you to define the language to be used on the workstation

.

B. Phoenix Reporting

This file is used by the Reporting System SRC. Among other things, the contents of this file define the internet host location of the database Server and set the Manager SRC debug level.

1. Report INI File

a) [COMMON] Section

This section contains information necessary for starting the Reporting System.

(1) DB_SERVER = {*server_name*}
DB_USER = {*user_name*}
DB_USER_PASSWORD = {*user_password*}

DB_SERVER, DB_USER, and DB_PASSWORD define the information required to gain access to the PHOENIX database. The PHOENIX database is identified by the name assigned by the DB SERVER variable. The user ID defined by the DB_USER variable and password value associated with the DB_USER_PASSWORD variable must be supplied before permission is granted to access the database.

(2) DEBUG_LEVELS = {*level*}

DEBUG_LEVELS determines the detail level of diagnostic message reporting. The messages are reported in a log file and are a useful tool for identifying error conditions. If this parameter has no defined options, no messages of this type are written to the log file. If no parameter is defined, the log file remains empty. Any single option or combination of options may be used; use the pipe, |, separator between each option.

(a) ***ERROR****—reports any condition that is out of the ordinary.*

(b) ***TRACE****—reports the entrance or invocation of a function.*

(c) ***INFO****—reports an in-depth trace of the code being executed in a function.*

Caution—*Leaving the INFO or TRACE parameters on for any period of time will cause the log file to become extremely large. Care should be taken when enabling these options.*

(3) TEMPORARY_PATH = {*path*}

The TEMPORARY_PATH parameter defines the working directory for temporary files used by the Reporting System.

(4) WORKING_PATH = {*path*}
The WORKING_PATH parameter defines the working directory for temporary files used by the Reporting System.

(5) OUTPUT_PATH = {*path*}
The OUTPUT_PATH parameter defines the location where the actual report data is stored.

(6) PROFILE_PATH = {*path*}
The PROFILE_PATH parameter defines the location where the INI file(s) for the Reporting System is stored.

(7) RECURRING_PATH = {*path*}
The RECURRING_PATH parameter defines the location where the INI file(s) for recurring reports is stored.

(8) BANNER_PAGE = {YES, NO}
The BANNER_PAGE parameter allows you to print (yes) a Banner page as page 1 of every report, which includes general information about the report, such as the selection criteria entered on the Entry Form.

(9) UL = {YES, NO}
The UL parameter adds UL reports to the menu.

b) [RPTSRV] Section

This section contains information required by the Reporting System executables.

(1) WWWROOT_PATH = {path}
The WWWROOT_PATH parameter defines the location of the system directory.

(2) SPOOL_PATH = {path}
The SPOOL_PATH parameter defines where the spool directory is for system data.

(3) IMAGE_PATH = {path}
The IMAGE_PATH parameter defines the location of images used by the Reporting System.

(4) PAGESRC_PATH = {path}
The PAGESRC_PATH defines the full path where HTML screen pages are stored.

(5) MAX_REPORTS = {#}

The MAX_REPORTS parameter defines how many reports can be run at the same time.

(6) AUTOLOGOUT = {#}

The AUTOLOGOUT parameter defines the length of time (in seconds) to wait before logging off the workstation if no activity has occurred. A value of 0 (zero) results in no logout.

(7) DEBUG_LEVEL

DEBUG_LEVELS determines the detail level of diagnostic message reporting. The messages are reported in a log file and are a useful tool for identifying error conditions. See "Debug Levels" on page 99 for an in-depth description of these options. If this parameter has no defined options, no messages of this type are written to the log file. If no parameter is defined, the log file remains empty. Any single option or combination of options may be used; use the pipe, |, separator between each option.

(a) ***ERROR****—reports any condition that is out of the ordinary.*

(b) ***TRACE****—reports the entrance or invocation of a function.*

(c) ***INFO****—reports an in-depth trace of the code being executed in a function.*

Caution—Leaving the INFO or TRACE parameters on for any period of time will cause the log file to become extremely large. Care should be taken when enabling these options.

(8) LOG_FILE_MAX_SIZE = 1000000

LOG_FILE_MAX_SIZE allows you to specify the maximum size in bytes of the Debug log files. Default value is 1000000. As the initial log file (report.log) reaches the size limit, Phoenix closes it and creates a new file (report_1.log) and writes to it. The digit suffix is incremented with each log file created.

(9) LOG_FILE_MODE = {a+, w+}

LOG_FILE_MODE allows you to specify how the Application Server initially opens and writes to the log. **a+** is the default. It opens a file for appending and creates the file if it does not exist. **w+** opens a file for writing. If the given file exists, its contents are destroyed.

c) HTML Section

The HTML Section defines the colors used in the Reporting System.

(1) BACKGROUND_COLOR = {hex}

The BACKGROUND_COLOR parameter defines the default color of the background in standard hexadecimal format.

(2) TEXT_COLOR = {hex}

The TEXT_COLOR parameter defines the default color of the text in standard hexadecimal format.

(3) LINK_COLOR = {hex}

The LINK_COLOR parameter defines the default color of links in standard hexadecimal format.

(4) VLINK_COLOR = {hex}

The VLINK_COLOR parameter defines the default color in standard hexadecimal format.

(5) ALINK_COLOR = {hex}

The ALINK_COLOR parameter defines the default color in standard hexadecimal format.

2. Paging Settings for Different Browsers

a) PAGESETUP.INI

Using the default ***PAGESETUP.INI*** is the first step for Phoenix Reporting to page correctly. This file is located in the ***"\phoenix\profiles\reports"*** folder on the machine where the Phoenix Reporting system is installed. The default PAGESETUP.INI is as follows:

(1) Setting up INI file

LPI	= 7
CPI	= 12
WIDTH	= 8.5
LENGTH	= 11
TOP_MARGIN	= 1.0
BOTTOM_MARGIN	= 1.0
LEFT_MARGIN	= 0.5
RIGHT_MARGIN	= 0.5
STYLE	= TEXT
FRIST_ADDRESS_LINE	= 13
LAST_ADDRESS_LINE	= 20
INDENT_ADDRESS	= 5

(2) Testing settings

Log into Phoenix Reporting, run a test report. After the report is generated, click on Print Preview and step through the pages to make sure paging is correct. There may be additional setups required to complete the process for Phoenix Reports to page correctly.

b) Microsoft Internet Explorer 7.0 and greater page setup

The Page Setup Option is located in the Print drop down on the right hand side of the Explorer Window. It can also be found in the File – Page Setup drop down on the top left hand side of the Explorer Window.

(1) Changing Page Setup Options

In Page Setup, start with these following options:

Paper Options:

Page Size: Select Letter
Select Portrait
Select Enable Shrink-to-Fit

Margins (inches):

Left : 0.75
Right : 0.75
Top : 0.75
Bottom : 0.75

(2) Testing the Settings

Log into Phoenix Reporting, run a test report. After the report is generated, click on Print Preview and step through the pages to make sure paging is correct.

If the report indicates that the heading/footer creeps, try changing ***"Shrink to Fit"*** setting to another value (i.e. change to 95%).

Unfortunately, there is no global value for the setting. It looks like it will be dependent on the printer selected for output. Once this value is modified, the report refreshes and report paging can be checked again.

c) Mozilla Firefox

Page Setup can be located in the File – Page Setup drop down on the top left hand side of the Browser Window.

(1) Changing Page Setup Options

In Page Setup, start with these following options.

Format & Options tab

Orientation: Select Portrait
Select Shrink to fit Page Width

Margins & Header/Footer tab

Margins (inches)
Top : 0.8
Bottom : 0.9
Left : 0.7
Right : 0.7

(2) Testing the Settings

Log into Phoenix Reporting, run a test report. After the report is generated, click on Print Preview and step through the pages to make sure paging is correct.

As with IE, if print preview indicates that the reporting paging is off, try changing the Scale Value from ***"Shrink To Fit"*** to another value.

d) Google Chrome

This web browser is not recommended for a couple of reasons:

1) There is no Print Preview option
2) All the Phoenix web pages (menus and entry forms) are left justified making the pages look off-centered. The reports still generate, menus just look strange.

V. System Monitoring

System Monitor is an APPSRV process that runs every minute. This is the same process that is responsible for moving expired Wait Events back to Pending, alerting operators regarding old events, HASP key is missing messages and so on.

A. There are 5 new tasks assigned to System Monitor

If System Monitor finds an issue with any of these items, a signal will be generated and submitted for processing to alert operators of the problem situations. The signals that are created will always have a Transmitter of PHOENIX (with exception of Zone 1 signals). The Zone ID indicates the type of problem System Monitor has uncovered.

Signal ID	Zone ID	Problem
<recv ID>	1	Receiver Comm Failure, collect has stopped sending signals
system	2	Low Disk Space, a drive has less and 20% free space

system	3	ABMsignal not purging, old signals in signal table
system	4	Relay Client not found, relay client has not connected
system	5	Relay Comm Failure, relay client stopped communicating
systemLog	20	ABMsignal purge success, shows number of rows deleted

1. Collect Monitoring

When System Monitor runs, it will check the Client Queue for collects which are connected to APPSRV. System Monitor will find the difference between the 'LastActionTime' and the current time. If the difference between the 2 values is greater than the value found for COLLECT_POLLING_INTERVAL (expressed in minutes) found in APPSRV.ini, a signal will be populated as follows and sent for processing:

a) Collect Monitoring Signal

Transmitter ID: PHOENIX
Signal ID: <recv ID> from collect.ini (user_name seen in Client Queue)
Zone ID: 1
Relatedinfo: Last signal received on <mm/dd/yyyy hh:mm:ss>, Polling interval: <mm>

b) Polling Interval

The 'Polling Interval' is taken from the APPSRV.ini setting for COLLECT_POLLING_INTERVAL.

If the problem persists after the COLLECT_POLLING_INTERVAL has elapsed again, another signal will be generated.

c) Polling Interval Value

Currently, there is only 1 polling interval value used for all collects, so the best value for polling interval should be based on the receiver which has the least activity. In the future, this feature may have to be expanded to provide different polling intervals for each collect.

2. Disk Space Available Monitoring

APPSRV on each startup, and once every 24 hrs, will run a System Monitor check to select disk drives and check for available disk space. If a drive has less than 20% available, a signal will be populated as follows and sent for processing:

a) Disk Space Monitoring Signal

Transmitter ID: PHOENIX
Signal ID: system

Zone ID: 2
Relatedinfo: [Threshold <x>%] C: <y>% free

b) Disk Space Values

The value in [ ] is the threshold APPSRV is using to decide with a problem exists or not. Any value equal or less than the threshold will generate a signal. The default value is 20%. This value can be changed with an entry to the APPSRV.ini.

In the [SYSTEM_MONITOR] section of APPSRV.ini, set MINIMUM_DISK_SPACE equal to the percentage override. For example, if a user wants to be notified when available disk space is equal or less than 10% - enter the following:

MINIMUM_DISK_SPACE = 10

***APPSRV** would have to be **stopped and restarted** for this value to take effect.*

3. ABMSignal Table Monitoring

ABMsignal table is purged automatically on regular bases. APPSRV on each startup, and once every 24 hours, will run a System Monitor check to do a count of the ABMsignal table. If there are more than 100 rows with a signal_date older than the number of days set for SIGNAL_PURGE_DAYS + 25%, a signal will be populated as follows and sent for processing:

a) Signal Table Monitoring Signal

Transmitter: PHOENIX
Signal: system
Zone: 3
Relatedinfo: <x> rows older than <y> days in ABMsignal table

b) Signal Table Values

The default setting for SIGNAL_PURGE_DAYS is set at 60. When System Monitor performs this check and finds over 100 rows with a 'signal_date' older than 75 days (60 + 25%), the signal will be created.

c) Signal Table Checks

This check was added because APPSRV was modified in X.4.1.3 to not purge ABMsignal table. There were no checks to see of the SQL job, which purged ABMsignal, was running on regular bases. Now that APPSRV is purging ABMsignal table, this System Monitor check is less important.

4. RELAY Monitoring

a) RELAY Client - PART I

If the network setup requires a Relay Client to run so that remote sites can receive and send messages, System Monitor can warn operators if a Relay Client has not connected to APPSRV.

(1) Activation

This feature can be activated or de-activated without having to stop/restart APPSRV. To activate this feature, insert an entry into the ABMzone table with the following data:

Zone ID: 4
Transmitter ID: PHOENIX
Sigtype ID: -1
Zone Name: Relay Client Not Found

(2) Event Alert

Adding this entry into the ABMzone table will alert System Monitor to check for a Relay Client in the Client Queue. The first check for a Relay client will begin 10 minutes after APPSRV has started to allow the remote clients' time to log on. If no Relay client is found to have connected, then a signal is populated as follows and sent for processing:

Transmitter ID: PHOENIX
Signal ID: system
Zone ID: 4
Relatedinfo: ABMzone flagged for relay, a Relay client connection is required

(3) De-activation

This signal will be generated each time System Monitor runs, which is once a minute, until a Relay client connects; or the entry from the ABMzone table is deleted.

b) RELAY Client- PART II

Whether or not the ABMzone entry is present, when Relay Client makes a connection to APPSRV, it inserts an entry into the Client Queue. When System Monitor comes across this entry it will find the difference between the 'LastActionTime' and the current time.

(1) Event Alert

If this difference is greater than 1 minute, a signal will be populated as follows and sent for processing:

Transmitter ID: PHOENIX
Signal ID: system
Zone ID: 5
Relatedinfo: Start Relay process on <hostname>

(2) Network Issues

There may have been a network issue which may have prevented the Relay Client from keeping constant

communication with APPSRV. Stopping and restarting process on the remote machine will re-establish a connection between the Relay Client and APPSRV.

5. ABMsignal Purge

ABMsignal table is purged a regular bases to keep collects and reporting processes running smoothly. The system defaults for purging are set to purge at every 60 minutes and keeping 60 days of signals in the ABMsignal table. However, these values can be overridden in the [SYSTEM_MONITOR] section of the APPSRV.ini.

a) INI Settings

SIGNAL_PURGE_DAYS = <number of days to keep in ABMsignal table>
SIGNAL_PURGE_FREQ = <how often to run a purge, expressed in minutes>

b) Event Signal

If any rows are deleted when System Monitor performs the purging process, a signal will be populated as follows and sent for processing:

Transmitter ID: PHOENIX
Signal ID: systemLog
Zone ID: 20
Relatedinfo: cutoff [<x>] days; Elapsed time[<mm:ss>]; rows [<y>]

Cutoff *refers to the value set for SIGNAL_PURGE_DAYS; elapsed time is how long the purge took, and rows display how many items were deleted. The 'systemLog' signal should be a non-event generating signal.*

B. ITEMS OF INTEREST

1. SYSTEM MONITOR CHECKS

The checks for Disk Space Monitoring and purging of ABMsignal are only performed at each APPSRV startup and only once a day thereafter. There did not seem to be a need to perform these checks more than once per day; however as in everything else, this can be overridden.

a) INI Setting

In the [SYSTEM_MONITOR] section of APPSRV.ini, set DISK_CHECK_INTERVAL = to the number of minutes between checks.

DISK_CHECK_INTERVAL =

Default is 1440 minutes (1440/60 = 24 hrs). To check every 12 hours set value equal to 720, to check every 48 hours set value 2880. If this value is modified, APPSRV will have to be stopped and restarted.

2. BENEFITS TO SYSTEM MONITOR GENERATING SIGNALS

a) Future Reference

The benefit of logging the information for future reference.

b) Specific Setup

Users can setup specific system wide instructions for the specific zones to advise the operators on how they should handle the systems errors being received.

E***xample***:

1) Set up an 'instruction/contact' Class with Classifier ID of 'internal'.
2) Then create a contact for Transmitter ID 'PHOENIX' and classifier of 'Internal'.
3) The Contact can be a shift supervisor.
4) Next create instructions for Transmitter ID of 'PHOENIX', sigtype of '-1', Call Classifier of 'internal', with more detailed explanation of error and what should be done.

C. Database Table Reference

1. ABMtransmitter Table

TransmitterID	DealerID	OrganizationID	SubscriberID	SiteID	Name
PHOENIX	-1	-1	-1	-1	Phoenix System

2. ABMzone Table

ZoneID	Transmitter	SigtypeID	ZoneName
1	PHOENIX	-1	Receiver Comm Failure
2	PHOENIX	-1	Low Disk Space
3	PHOENIX	-1	ABMsignal not purging
4	PHOENIX	-1	Relay Client not found
5	PHOENIX	-1	Relay Comm Failure
20	PHOENIX	-1	ABMsignal purge count

3. ABMsigtype Table

SignalID	Event Flag	Description
system	Y	Phoenix generated signal
systemLog	N	Phoenix generated signal - log only

VI. Backup, Archive and Purge Processes

Backups—A frequent, routine save of all data in the database and the database structure. You should be able to fully restore data from a successful backup. Backups are generally recycled; they are not saved forever.

Archives—An archive is a periodic save of the data in the database. It provides long-term storage for retrieving historical data. Archives are generally saved indefinitely.

Purges—A purge is a periodic deletion of data from the database based on certain criteria, such as all records in the Signal table with a Signal Create Date less than a specific date. If you do not periodically purge data, the tables become unmanageably large, resulting in poor system performance.

A. There are three methods of backing up and archiving a database:

1. **DB Utility**
   Use the utility that comes with the database

2. **ABM DBA Utility**
   Use ABM's **dba** utility for archiving and purging the database based on specified criteria, such as date and table name(s)

3. **SQL Script**
   Write your own SQL scripts for ad hoc backups with the option of using WinAT Scheduler.

For routine, periodic backups of the database, ABM recommends that you use the utility that comes with your database system. See the database documentation for instructions on how to perform backups. (These utilities do not offer purge capabilities; they do not delete the data from your database.)

a) To archive the database using ABM's utility:

dba *is a database purging and archiving utility to use with the ODBC compliant database that is supported by Phoenix. You may archive to a file (you name it) or to a device, such as a tape drive.*

1. Go to Start menu, ***Programs***, click on ***Command Prompt***.
2. Change the path to the archive folder.

drive:\> cd phoenix\archives

3. Type the dba command string as explained below:

dba -cvf *file_name*.dba -L *server_name,user,user_password*

For example: To backup a database to a file called "latest.dba" on a server named Orion, type:

dba -cvf latest.dba -L orion,phoenix,flight

The following explains each of the parameters in the command:

dba database archive command
-c create a new archive file

-v	verbose mode (lists each table backed up and the number of records in each)
-f	filename to perform the action on latest.dba name of new file to be created; it **must** have a .dba extension
-L	login parameters necessary to connect to the database server
orion	server name
phoenix,flight	valid user and password for the server

All of the above parameters are required except for the **v**, which is optional.

VII. Backup and Archive Procedures

Administrators are tasked with many responsibilities including network, database, and user maintenance. While the duties they perform have various priorities, one of the most important and generally forgotten is the archiving of valuable data. ABM highly recommends that backup procedures be defined prior to the implementation of Phoenix. This ensures that when a problem arises, a contingency plan including the restoration of required data will help to minimize lost data or down time.

For this discussion on backups this section focuses on the Phoenix directory only. As noted, administrators should develop a backup procedure for other critical files.

A. Data Archiving (Backup)

In order to properly recover a damaged or lost data store, an administrator must be able to restore a copy of the data from some form of recording media.

This could include: tape car tridge, floppy disks, or even CD-ROM.

Generally there are utilities that allow the administrator to perform data archives:

- Windows NT has a *backup* utility that can backup specified files and directories.
- Most RDBMS offer a database archive utility.
- The Phoenix dba utility facilitates simple archives of Phoenix data stores.

VIII. Security

Security should always be evaluated based on “the type of security that is needed for the working environment” not necessarily by which one has more security features. By using the features of Windows and Phoenix, an installation site is able to maximize protection against security violations without giving the appearance of a restrictive, user unfriendly system.

Most security starts with system users having to identify themselves to the operating system or applications through a process know as an *authentication* or more commonly *logon*. This is accomplished by providing a valid account name and password. Once a

user is granted access, the system sets the permissions and rights that allow him or her to access files, programs, and shared resources.

A. Windows Security

Windows meets the security rating of C2; a rating developed by the US Government in cooperation with the National Computer Security Center (NCSC) and the National Institute of Standards and Technologies (NIST). The rating is established to give the user a workable level of security with the option of tailoring the security features as needed. Windows security can be sorted into three categories: *allowing access into the system* (account), *ability to perform various tasks* (user rights), and *what was done on the system* (audit). Since Windows' security is built into the core of the operating system, the administrator needs only to decide how much security is needed and then utilize that function.

1. Account

By definition the user account is the basis of security; without a valid Login ID and Password a user is unable to gain access to a system or programs. To ensure a high level of security at the operating system level, the administrator should ensure the following:

- Assign each user an individual account for authentication
- Assign each user an individual password
- Require that all passwords are changed on a regular basis
- Require that all passwords are a minimum of six alpha-numeric characters in length with at least one alpha character and one numeric character
- Require that Passwords or ID's are not shared between individuals

2. User Rights

User rights allow an administrator to grant policies to groups and users. A **right** authorizes a user to perform certain actions under their account. Rights are assigned based on system level function and on the user's permissions. Rights include but are not limited to:

- The ability to perform system backups
- The ability to shutdown the system
- The ability to create and share system files

3. Auditing

Security auditing is used to monitor for any potential security violations or areas that may need greater security attention. Some of the auditing options available are:

- Auditing of users logon/logoff activity
- Monitoring of File/Object access
- Startup/Shutdown activity

B. Phoenix Security

By itself, Windows offers one of the most secured systems available today. Phoenix takes this layer of security and builds upon it. In order to utilize Phoenix applications, a user must present their Phoenix System Login ID and a Password; this prevents unauthorized users from accessing Phoenix applications. Once a user is validated as a Phoenix user, their account configuration determines access to the various components and data stores of the system. The Phoenix system security layer is implemented using the User and ClassAuth tables as well as the configuration of various Initialization (.ini) files.

1. User

In Phoenix, the User table establishes and validates Phoenix system users. To ensure a higher level of security, the administrator should establish the following guidelines:

- Give each user an individual ID for authentication purposes
- Give each user a unique Password
- Require users to change passwords in a predefined cycle
- Configure Authorization Levels to deliver access on an "as needed" basis
- Set effective date and time and expiration date and time for User IDs appropriately

a) Predefined Authorization Levels

The Phoenix system is pre-installed with seven predefined Authorization Levels:

Level	**Function**	**Restrictions** (cannot access)
1	Alarm Processing Search	Special Schedules No Action Clear Pending Some tables restricted*
2	Alarm Processing Search	No restrictions Some tables restricted*
3	Alarm Processing Data Entry Reporting Search	No restrictions No Delete (can read, write) No restrictions No restrictions
4	Alarm Processing Data Entry Search Reporting Manager	No restrictions No restrictions No restrictions No restrictions No restrictions
5	Voice Response	
6	Alarm Processing Search	No Clear Pending Some tables restricted*

7	Alarm Processing Search Reporting	No Clear Pending Some tables restricted* No restrictions

* Users with this Access Level cannot access the following tables:

Access Control	Log	Service Types	Category	Message
Class		Packet Type	Sigclass	Classauth
Preference		Sigcat	Sigcontrol	Component
Types		Process	Sigtypes	Counter
Receiver		Table Types	Device	Remote
User		Group	Resolution	

b) ClassAuth

You can set up custom Authorization Levels using the Classauth table. The Classauth table defines levels or classes of security authorizations. When authorization levels are set properly, a user may subscribe to multiple levels of security.

The levels of authorization available within the ClassAuth table are:

- Execute permissions for each Phoenix service (Data Entry, Alarm Processing, etc.)
- Execute permissions for each individual component of each Phoenix service (SQL Administration, etc.)
- Execute, Read, Write, and Delete permissions for each physical database table (Dealer, Subscriber, Sigtype, etc.)

(1) Authorization Level

When entering users, the *authorization_level* column of the table defines the user's level of access. Many users may share the same authorization level.

This authorization level column of the User table establishes a relationship between the user and multiple row entries in the Classauth and User tables. This value identifies multiple rows of data with various *types* and *id* column values. Each Classauth entry defines the types of authorization as:

- Type '1' specifies a Classauth Service entry
- Type '2' specifies a Classauth Component entry
- Type '3' specifies a Classauth Table entry Users' levels of access are defined by Type.

(2) Service Entries

For a Service entry, Type **1**, there are multiple *ID* column values that specify a particular Service. The current available Services and their numeric values are as follows:

The Execute_Flag column for each of these service entries indicates the service's accessibility. An Execute_Flag value of **1** means that the service is accessible (on) whereas an Execute_Flag value of **o** means that the service is inaccessible (off).

(3) Component Entries

A Classauth entry having the column value for Type equal to **2** is a Component. A Component is defined as a function or subprocess of a Service. A Component performs a specific task or operation within a given Service. For a specific authorization level, there would normally be many Component entries, one for each Component of every Service in the Phoenix system. The *ID* column value for each of these table entries uniquely identifies particular service Components for which access permission is defined. By setting the execute permission for each and every Component listed in the table within a given authorization level, one is specifying what features or functions of a service that a specific user may be able to perform.

As is the case throughout Phoenix, the marker value (-1) in a database column is a 'wildcard' or 'match-all' entries. By entering a **-1** in the ID column for an Authorization Level of Type Component, one is granting access permission to ALL components.

To see a complete list of all Components in your Phoenix system and their numeric values, grouped together by Service, in Data Entry, open the Component Type table, click on the **Q**UERY tool, and then click on the **L**IST **V**IEW tool.

Each set of Components, grouped by Service type, are identified by a numeric value. Components by Service Type are organized such that they span a range of numbers as indicated above. The values for Service, Component, and Table Types cannot be changed.

(4) Table Entries

A Classauth entry having the column value for Type equal to **3** is a Table Type classauth. For a specific authorization level, there would normally be many tables. The *id* column value for each row of a Table identifies the particular table for which an access level is being defined. As in Component entries, a -1 value entry in the *id* field specifies ALL tables. Assigning a **1** (on) of a **0** (off) entry in the Read or Delete columns permits or denies that particular access permission. The Execute column is not applicable to table oriented permission.

For example:

Auth Level	Type	ID	Exec Perm	Read Perm	Write Perm	Delete Perm
1	1 (service)	0 (Collect)	1 (on)	0 (off)	0 (off)	0 (off)
1	1 (service)	1 (Browse)	1 (on)	0 (off)	0 (off)	0 (off)
1	1 (service)	2 (Alarm)	1 (on)	0 (off)	0 (off)	0 (off)
1	1 (service)	3 (Maint)	1 (on)	0 (off)	0 (off)	0 (off)
1	1 (service)	4	1 (on)	0 (off)	0 (off)	0 (off)
1	2	-1 (all)	1 (on)	0 (off)	0 (off)	0 (off)
1	3 (table)	-1 (all)	0 (off)	1 (on)	1 (on)	1 (on)

*This set of entries for the default class authorization defines the access permission to the five Services: Collect, Alarm, Browser, Maintenance, and Manager. All are executable by the user. The **-1** entry in the ID column for Type 2, Component Type, indicates that all Components of each Service are accessible. Also, the **-1** entry in the ID column for Type 3, Table Type, indicates that Read, Write, and Delete operations are permitted on all tables of the database.*

IX. Trouble-shooting

This section answers some commonly asked questions regarding the Phoenix system.

Why doesn't the Phoenix Application Service start automatically upon reboot?
Go to the Start menu, Control Panel, Services option (Windows NT) or Start menu, Control Panel, Administrative Tools, Services option (Windows 2000) and click the Phoenix Application Server (**Appsrv**), choose the Startup button, then select Automatic.

When entering Alarm Processing, Browser, or Data Entry, I receive the following message: "error connecting to Appsrv listener service –13, 10061".
Go to Start menu, Control Panel, Services option and verify that the Phoenix Application Server (**Appsrv**) is started. If it is not running, start it and try logging in to Alarm Processing, Browser, or Data Entry again. If Appsrv is already running, check the profile for the application and insure that the APP_SERVER variable is set to the internet host name of the machine where Appsrv is running.

When entering Alarm Processing, Browser, or Data Entry, I receive the following message: "can not find 'Application Profile'".
Verify that the application shortcut contains the full path of the related application profile.

When querying the Data Entry database I receive the following message: "Row count XXX exceeds maximum".
This user-definable parameter within the maint.ini allows the administrator to set the maximum number of rows that may be returned by a single Query. Specifying a more detailed Query may also resolve the error message.

Caution—*Contact ABM technical support before changing this value.*

When attempting to log into the system I receive the following message: "User ID xxx has expired".
This is a security feature that allows the administrator to define an expiration date and time after which a user is no longer allowed access to the system. Contact your system administrator.

When logging into the system I receive the following message: "error invalid password".
The password you have entered is invalid. Retype the password making sure each character is correct. Since passwords are case sensitive, it is important to use the proper case when logging in; make sure the *Caps Lock* key is not on. If there continues to be a problem, contact your system administrator.

When using MicroSoft Internet Browser to address the *Phoenix Reporting System*, I receive the following error message when entering the path to the Phoenix report writer's home page: "HTTP/1.0 404 Object Not Found"
This message indicates that an incorrect path name was entered for the Reporting System home page on the server machine. Check the location of the home page and reenter.

X. Glossary

Administrative User
A user with advanced knowledge capable of administering a complex computer system.

Active
An event which is currently being processed by an operator.

Application Server (Appsrv)
The hub of the Phoenix System, the system manager of Phoenix.

Column
A relational term for a ***field***.

Database
A collection of related objects (including tables, forms, reports, queries, and scripts) created and organized by a database management system.

Database Management System (DBMS)
Application software that controls the data in a database, including overall organization, storage, retrieval, security, and data integrity.

Field
A place in a record preserved for a specific piece of information.

Foreign Components
Foreign components talk to Phoenix through ODBC, as opposed to native components which are part of the Phoenix system.

Hex (Hexadecimal)
The base-16 numbering system that represents the binary numbers computers use internally.

Inheritance
The ability of one object to pass on properties to another object.

Initial Inheritance
Inheritance at the time a new record is created.

Marker Record
A placeholder record which completes a relationship necessary to the data architecture.

Marker Value
A field entry having a value ***-1***.

Native Components
Native components are part of the Phoenix system, as opposed to foreign components which talk to Phoenix through ODBC

Open Database Connectivity (ODBC)
Microsoft's strategic interface for accessing data in a heterogeneous environment of relational and non-relational database management systems.

Pending
An event or signal which requires examination by an operator.

Phoenix System
The combination of all components which work together to provide all aspects of security monitoring.

Primary Key
The column or combination of columns that defines a record as unique.

Relational Database
A database which addresses its data constructed upon the relationship that exists between records.

Relational Inheritance
Inheritance resulting from the association of one object to another.

SPC's (Service Provider Components)
A group of Appsrv threads whose exclusive purpose is to provide data services for interested SRC programs.

SRC's (Service Requester Components)
User interfaces that provide a user with a window into the operating system applications environment.

System Directory
System director is another term for the Phoenix application directory.

Table
The structured organization of data in a database; each table is composed of rows and columns.

Token
Token is the smallest language unit which conveys a meaning. Symbol is a synonym for the word token. In written text, words are symbols and tokens. In analyzing the meaning of a written phrase or statement, the source language text is subdivided into tokens and evaluated based upon the arrangement and presence or absence of various known tokens.

Working Directory
The current directory to which all files are saved; the system assumes this is the current path unless otherwise noted.

XI. NOTES: